

502/0010500

日本国特許庁  
JAPAN PATENT OFFICE



別紙添付の書類に記載されている事項は下記の出願書類に記載されて#4  
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed  
with this Office

出願年月日

Date of Application:

2001年 1月10日

出願番号

Application Number:

特願2001-002221

出願人

Applicant(s):

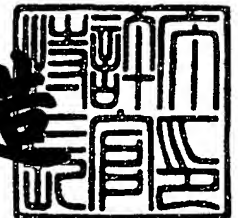
ソニー株式会社

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2001年12月14日

特許庁長官  
Commissioner,  
Japan Patent Office

及川耕造



SON-2321

PATENT APPLICATION

#4  
4-24-03  
JM

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Patent Application of )  
HIDEAKI WATANABE ET AL ) APPLICATION BRANCH  
Serial No. To be assigned )  
Filed: January 9, 2002 )  
For: PUBLIC KEY CERTIFICATE ISSUING )  
SYSTEM, PUBLIC KEY CERTIFICATE )  
ISSUING METHOD, DIGITAL )  
CERTIFICATION APPARATUS AND )  
PROGRAM STORAGE MEDIUM )

10/040436  
01/09/02

CLAIM TO PRIORITY UNDER 35 USC 119

Commissioner for Patents  
Washington, D.C. 20231

Sir:

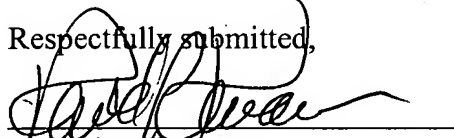
The benefit of the filing date of the following prior application filed in the following foreign country is hereby requested and the right of priority provided under 35 U.S.C. 119 is hereby claimed:

Japanese Patent Appl. No. 2001-002221 filed January 10, 2001

In support of this claim, filed herewith is a certified copy of said original foreign application.

Date: January 8, 2002

Respectfully submitted,

  
Ronald P. Kananen  
Registration No. 24,104

**RADER, FISHMAN & GRAUER, PLLC**  
Lion Building  
1233 20<sup>th</sup> Street, N.W.  
Washington, D.C. 20036  
Tel: (202) 955-37650  
Customer No. 23353

【書類名】 特許願

【整理番号】 0000579506

【提出日】 平成13年 1月10日

【あて先】 特許庁長官殿

【国際特許分類】 H04L 9/32

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社  
内

【氏名】 渡辺 秀明

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社  
内

【氏名】 石橋 義人

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社  
内

【氏名】 松山 科子

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社  
内

【氏名】 二村 一郎

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社  
内

【氏名】 昆 雅士

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社  
内

【氏名】 岡 誠

【特許出願人】

【識別番号】 000002185  
【氏名又は名称】 ソニー株式会社  
【代表者】 出井 伸之

【代理人】

【識別番号】 100101801  
【弁理士】  
【氏名又は名称】 山田 英治  
【電話番号】 03-5541-7577

【選任した代理人】

【識別番号】 100093241  
【弁理士】  
【氏名又は名称】 宮田 正昭  
【電話番号】 03-5541-7577

【選任した代理人】

【識別番号】 100086531  
【弁理士】  
【氏名又は名称】 澤田 俊夫  
【電話番号】 03-5541-7577

【手数料の表示】

【予納台帳番号】 062721  
【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1  
【物件名】 図面 1  
【物件名】 要約書 1  
【包括委任状番号】 9904833

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 公開鍵証明書発行システム、公開鍵証明書発行方法、および情報処理装置、情報記録媒体、並びにプログラム記憶媒体

【特許請求の範囲】

【請求項 1】

公開鍵証明書を利用するエンティティの公開鍵証明書を発行する認証局と、  
管轄エンティティから受領する公開鍵証明書発行要求を前記認証局に対して送信する登録局とを有し、

前記認証局は、各々が異なる署名方式を実行する複数の認証局によって構成され、前記登録局からの公開鍵証明書発行要求に応じて、前記複数の認証局間で公開鍵証明書を転送し、各認証局において異なる署名アルゴリズムに従って公開鍵証明書を構成するメッセージデータに対する電子署名を実行し、異なる署名アルゴリズムに従った複数の署名を格納した複数署名付き公開鍵証明書を発行する構成を有することを特徴とする公開鍵証明書発行システム。

【請求項 2】

前記複数の認証局は、

R S A 署名アルゴリズムに従った署名生成処理を実行する R S A 認証局と、  
楕円曲線暗号 (E C C) 署名アルゴリズムに従った署名生成処理を実行する E C C 認証局とを含み、

前記複数署名付き公開鍵証明書に格納された署名は、R S A 署名アルゴリズムに従った署名と、楕円曲線暗号 (E C C) 署名アルゴリズムに従った署名とを含むことを特徴とする請求項 1 に記載の公開鍵証明書発行システム。

【請求項 3】

前記複数の認証局の少なくとも 1 以上の認証局は、

公開鍵証明書の拡張領域に生成した署名、および生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行する構成を有することを特徴とする請求項 1 に記載の公開鍵証明書発行システム。

【請求項 4】

前記複数の認証局の少なくとも 1 以上の認証局は、

公開鍵証明書の基本領域および拡張領域以外の領域に生成した署名を格納し、拡張領域に生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行する構成を有することを特徴とする請求項 1 に記載の公開鍵証明書発行システム。

【請求項 5】

前記複数の認証局の少なくとも 1 以上の認証局は、

公開鍵証明書に 2 以上の署名が含まれるか否かを示すフラグ情報を公開鍵証明書に格納する処理を実行する構成を有することを特徴とする請求項 1 に記載の公開鍵証明書発行システム。

【請求項 6】

公開鍵証明書を利用するエンティティの公開鍵証明書を発行する認証局と、管轄エンティティから受領する公開鍵証明書発行要求を前記認証局に対して送信する登録局とを有し、登録局からの要求に応じて公開鍵証明書を発行する公開鍵証明書発行方法において、

前記認証局は、各々が異なる署名方式を実行する複数の認証局によって構成され、

前記登録局からの公開鍵証明書発行要求に応じて、前記複数の認証局間で公開鍵証明書を転送し、各認証局において異なる署名アルゴリズムに従って公開鍵証明書を構成するメッセージデータに対する電子署名を実行し、異なる署名アルゴリズムに従った複数の署名を格納した複数署名付き公開鍵証明書を発行することを特徴とする公開鍵証明書発行方法。

【請求項 7】

前記複数の認証局の少なくとも 1 以上の認証局は、

署名の付加された公開鍵証明書に対して、付加された署名と異なる署名アルゴリズムを適用して署名を生成して前記公開鍵証明書に付加する署名ステップを実行することを特徴とする請求項 6 に記載の公開鍵証明書発行方法。

【請求項 8】

前記複数の認証局は、

R S A 署名アルゴリズムに従った署名生成処理を実行する R S A 認証局と、

楕円曲線暗号（ECC）署名アルゴリズムに従った署名生成処理を実行する ECC 認証局とを含み、

RSA 認証局において、RSA 署名アルゴリズムに従った署名生成処理を実行し、

ECC 認証局において、ECC 署名アルゴリズムに従った署名生成処理を実行し、

RSA 署名アルゴリズムに従った署名と、楕円曲線暗号（ECC）署名アルゴリズムに従った署名とを含む複数署名付き公開鍵証明書を発行することを特徴とする請求項 6 に記載の公開鍵証明書発行方法。

【請求項 9】

前記複数の認証局の少なくとも 1 以上の認証局は、

公開鍵証明書の拡張領域に生成した署名、および生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行することを特徴とする請求項 6 に記載の公開鍵証明書発行方法。

【請求項 10】

前記複数の認証局の少なくとも 1 以上の認証局は、

公開鍵証明書の基本領域および拡張領域以外の領域に生成した署名を格納し、拡張領域に生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行することを特徴とする請求項 6 に記載の公開鍵証明書発行方法。

【請求項 11】

前記複数の認証局の少なくとも 1 以上の認証局は、

公開鍵証明書に 2 以上の署名が含まれるか否かを示すフラグ情報を公開鍵証明書に格納する処理を実行することを特徴とする請求項 6 に記載の公開鍵証明書発行方法。

【請求項 12】

公開鍵証明書の検証処理を実行する情報処理装置であり、

公開鍵証明書内の基本領域および拡張領域に格納された署名情報に記録された複数の署名アルゴリズム中から、自己の情報処理装置において検証処理可能な署名アルゴリズムを選択して、該選択された署名アルゴリズムに従った署名の検証

処理を実行する構成を有することを特徴とする情報処理装置。

【請求項 1 3】

公開鍵証明書を検証処理を実行する情報処理装置であり、  
複数の署名アルゴリズムに従った署名検証処理機能を備えたことを特徴とする  
情報処理装置。

【請求項 1 4】

前記複数の署名アルゴリズムは、  
R S A 署名アルゴリズムと、楕円曲線暗号（E C C）署名アルゴリズムとを含む  
ことを特徴とする請求項 1 3 に記載の情報処理装置。

【請求項 1 5】

公開鍵を格納した公開鍵証明書を格納した情報記録媒体であり、  
前記公開鍵証明書は、複数の署名アルゴリズムに従った署名を格納した構成で  
あることを特徴とする情報記録媒体。

【請求項 1 6】

前記複数の署名アルゴリズムは、  
R S A 署名アルゴリズムと、楕円曲線暗号（E C C）署名アルゴリズムとを含む  
ことを特徴とする請求項 1 5 に記載の情報記録媒体。

【請求項 1 7】

公開鍵証明書を利用するエンティティの公開鍵証明書を発行する公開鍵証明書  
発行処理をコンピュータ・システム上で実行せしめるコンピュータ・プログラム  
を提供するプログラム記憶媒体であって、前記コンピュータ・プログラムは、  
署名の付加された公開鍵証明書に対して、付加された署名と異なる署名アルゴ  
リズムを適用して第 2 の署名を生成して付加する署名ステップ  
を有することを特徴とするプログラム記憶媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は電子配信システムにおいて暗号化データ送信に使用される公開鍵の正  
当性を証明するための公開鍵証明書の発行処理に関する公開鍵証明書発行システ



ム、公開鍵証明書発行方法、および情報処理装置、情報記録媒体、並びにプログラム記憶媒体に関する。さらに、公開鍵証明書を発行する認証局（CA）において、複数の署名アルゴリズムに対応した公開鍵証明書を発行し、公開鍵証明書を利用するエンティティにおける利便性を高めた公開鍵証明書発行システム、公開鍵証明書発行方法、および情報処理装置、情報記録媒体、並びにプログラム記憶媒体に関する。

#### 【0002】

##### 【従来の技術】

昨今、ゲームプログラム、音声データ、画像データ、文書作成プログラム等、様々なソフトウェアデータ（以下、これらをコンテンツ（Content）と呼ぶ）が、インターネット等のネットワークを介して流通している。また、オンラインショッピング等、ネットワークを介した商品売買も次第に盛んになってきている。

#### 【0003】

このようなネットワークを介したデータ通信においては、データ送信側とデータ受信側とが互いに正規なデータ送受信対象であることを確認した上で、必要な情報を転送する、すなわちセキュリティを考慮したデータ転送構成をとるのが一般的となっている。データ転送の際のセキュリティ構成を実現する1つの手法が、転送データの暗号化処理、データに対する署名処理である。

#### 【0004】

暗号化データは、所定の手続きによる復号化処理によって利用可能な復号データ（平文）に戻すことができる。このような情報の暗号化処理に暗号化鍵を用い、復号化処理に復号化鍵を用いるデータ暗号化、復号化方法は従来からよく知られている。

#### 【0005】

暗号化鍵と復号化鍵を用いるデータ暗号化・復号化方法の態様には様々な種類があるが、その1つの例としていわゆる公開鍵暗号方式と呼ばれる方式がある。公開鍵暗号方式は、発信者と受信者の鍵を異なるものとして、一方の鍵を不特定のユーザが使用可能な公開鍵として、他方を秘密に保つ秘密鍵とするものである。例えば、データ暗号化鍵を公開鍵とし、復号鍵を秘密鍵とする。あるいは、認証

子生成鍵を秘密鍵とし、認証子復号鍵を公開鍵とする等の態様において使用される。

【0006】

暗号化、復号化に共通の鍵を用いるいわゆる共通鍵暗号化方式と異なり、公開鍵暗号方式では秘密に保つ必要のある秘密鍵は、特定の1人が持てばよいための鍵の管理において有利である。ただし、公開鍵暗号方式は共通鍵暗号化方式に比較してデータ処理速度が遅く、秘密鍵の配送、デジタル署名等のデータ量の少ない対象に多く用いられている。公開鍵暗号方式の代表的なものにはRSA (Rivest-Shamir-Adleman) 暗号がある。これは非常に大きな2つの素数（例えば150桁）の積を用いるものであり、大きな2つの素数（例えば150桁）の積の素因数分解（および離散対数）する処理の困難さを利用している。

【0007】

さらに、公開鍵暗号方式の代表的なものとして、楕円曲線暗号 (Elliptic Curve Cryptography (ECC)) を用いた方法がある。これは楕円曲線とよばれる曲線上の点の間で演算が定義でき、その上で、離散対数問題の類似物（楕円離散対数問題）が作成可能なことに基づいた方式である。

【0008】

素因数分解（および離散対数）に基づくRSA暗号方式は、準指数的な解読法を持つのに対して楕円離散対数は、指数的解読しか不可能とされており、離散対数問題に基づくRSA暗号方式の鍵サイズが512, 1024, または2048ビットとなるのに対して、楕円曲線暗号方式 (ECC) の鍵サイズは160, 192, または224ビット程度で同等の安全性が保持され、鍵サイズの短さによって処理速度を高めることが可能となる。

【0009】

公開鍵暗号方式では、不特定多数に公開鍵を使用可能とする構成であり、配布する公開鍵が正当なものであるか否かを証明する証明書、いわゆる公開鍵証明書を使用する方法が多く用いられている。例えば、利用者Aが公開鍵、秘密鍵のペアを生成して、生成した公開鍵を認証局に対して送付して公開鍵証明書を認証局から入手する。利用者Aは公開鍵証明書を一般に公開する。不特定のユーザは公

公開鍵証明書から所定の手続きを経て公開鍵を入手して文書等を暗号化して利用者 A に送付する。利用者 A は秘密鍵を用いて暗号化文書等を復号する等のシステムである。また、利用者 A は、秘密鍵を用いて文書等に署名を付け、不特定のユーザが公開鍵証明書から所定の手続きを経て公開鍵を入手して、その署名の検証を行なうシステムである。

## 【 0 0 1 0 】

公開鍵証明書について図 1 を用いて説明する。公開鍵証明書は、公開鍵暗号方式における認証局（CA : Certificate Authority または IA : Issuer Authority）が発行する証明書であり、ユーザが自己の ID、公開鍵等を認証局に提出することにより、認証局側が認証局の ID や有効期限等の情報を付加し、さらに認証局による署名を付加して作成される証明書である。

## 【 0 0 1 1 】

図 1 に示す公開鍵証明書は、証明書のバージョン番号、認証局（IA）が証明書利用者に対し割り付ける証明書の通し番号、上述の RSA、ECC 等の電子署名に用いたアルゴリズム、およびパラメータ、認証局の名前、証明書の有効期限、証明書利用者の名前（ユーザ ID）、証明書利用者の公開鍵並びに電子署名を含む。

## 【 0 0 1 2 】

電子署名は、証明書のバージョン番号、認証局が証明書利用者に対し割り付ける証明書の通し番号、電子署名に用いたアルゴリズムおよびパラメータ、認証局の名前、証明書の有効期限、証明書利用者の名前並びに証明書利用者の公開鍵全体に対し生成される電子署名であり、例えばハッシュ関数を適用してハッシュ値を生成し、そのハッシュ値に対して認証局の秘密鍵を用いて生成したデータである。

## 【 0 0 1 3 】

認証局は、図 1 に示す公開鍵証明書を発行するとともに、有効期限が切れた公開鍵証明書を更新し、不正を行った利用者の排斥を行うための不正者リストの作成、管理、配布（これをリボケーション : Revocation と呼ぶ）を行う。また、必要に応じて公開鍵・秘密鍵の生成も行う。

## 【0014】

一方、この公開鍵証明書を利用する際には、利用者は自己が保持する認証局の公開鍵を用い、当該公開鍵証明書の電子署名を検証し、電子署名の検証に成功した後に公開鍵証明書から公開鍵を取り出し、当該公開鍵を利用する。従って、公開鍵証明書を利用する全ての利用者は、共通の認証局の公開鍵を保持している必要がある。

## 【0015】

## 【発明が解決しようとする課題】

上述のような認証局発行の公開鍵証明書を用いた公開鍵暗号方式によるデータ送信システムにおいて、公開鍵証明書の電子署名を検証し、電子署名の検証に成功した後に公開鍵証明書から公開鍵を取り出して、例えば公開鍵暗号方式による認証処理、あるいは公開鍵暗号方式による転送データの暗号化処理、あるいは復号処理を実行することが可能となる。しかしながら、公開鍵暗号方式による様々な処理を実行するユーザデバイス等のエンティティは、前述したECC、RSA等の様々な暗号方式アルゴリズムのすべてに対応可能であることは少なく、ECCアルゴリズム、あるいはRSAアルゴリズムにのみ対応した処理が可能であるといった構成が多い。

## 【0016】

このような単独、あるいは特定の暗号アルゴリズムのみ処理可能なデバイスは、そのアルゴリズムに従った署名方式を持つ公開鍵証明書のみ利用可能となり、他の方式で署名された公開鍵証明書は受け取っても署名の検証が実施できず、公開鍵証明書の検証が不可能になるという事態が発生する。

## 【0017】

従来は、例えば図2に示すように、ECCアルゴリズムを処理可能なECCデバイス23は、ECCアルゴリズムによる署名処理を実行するECC登録局(ECC-RA: Registration Authority)22に公開鍵証明書発行要求、あるいは更新要求を行ない、ECC登録局22は、各サービスに参加するエンティティ、機器を認証し、各エンティティ、機器からの公開鍵の公開鍵証明書発行要求を受領し、これをECCアルゴリズムによる署名処理を実行するECC認証局(ECC

C-CA) 21に送信し、ECC認証局(ECC-CA) 21は、ECCアルゴリズムによる署名処理を実行した公開鍵証明書を発行してECC登録局22を介してECCデバイス23に配布する。

## 【0018】

一方、RSAアルゴリズムを処理可能なRSAデバイス33は、RSAアルゴリズムによる署名処理を実行するRSA登録局(RSA-RA:Registration Authority) 32に公開鍵証明書発行要求、あるいは更新要求を行ない、RSA登録局32は、各サービスに参加するエンティティ、機器を認証し、各エンティティ、機器からの公開鍵の公開鍵証明書発行要求を受領し、これをRSAアルゴリズムによる署名処理を実行するRSA認証局(RSA-CA) 31に送信し、RSA認証局(RSA-CA) 31は、RSAアルゴリズムによる署名処理を実行した公開鍵証明書を発行してRSA登録局32を介してRSAデバイス33に配布する。

## 【0019】

このように、異なる複数の署名方式に対応したそれぞれの処理系統を構築し、それぞれの処理系統によって構築されたシステム内で閉じられた公開鍵暗号方式による認証、暗号化データ通信が実行される。

## 【0020】

ECCデバイス23は、RSAデバイス33からRSA方式で署名が施されたRSAデバイス33の公開鍵証明書を受信しても署名検証を実行することができず、公開鍵証明書の正当性の検証が不可能になり、証明書としての機能を果たさない。またその逆にRSAデバイス33は、ECCデバイス23からECC方式で署名が施されたECCデバイス23の公開鍵証明書を受信しても署名検証を実行することができず、公開鍵証明書の正当性の検証が不可能になる。

## 【0021】

図2のECCデバイス23と、RSAデバイス33との間でそれぞれ相手方の公開鍵証明書の正当性の確認を行なうためには、それぞれが相手方から受信した公開鍵証明書をECC登録局22、RSA登録局32に送信し、さらに、ECC認証局(ECC-CA) 21、RSA認証局(RSA-CA) 31に送信し、E

CC認証局（ECC-CA）21、RSA認証局（RSA-CA）31との間で相互に問い合わせを実行し、その結果を各デバイスが受け取って認証の代わりとする方法をとらざる得ない。

【0022】

この構成を図3に示す。RSAデバイス33はECCデバイス23と相互認証を行うため、RSAデバイス33自身の公開鍵証明書をECCデバイス23に対して送信する。この送付される公開鍵証明書にはRSA認証局（RSA-CA）31により署名付けがなされている。ECCデバイス23はRSA認証局（RSA-CA）31が発行した証明書を検証することができないため、証明書の有効性をECC登録局（ECC-RA）22を通してECC認証局（ECC-CA）21へ問い合わせる。

【0023】

ECC認証局（ECC-CA）21はRSA認証局（RSA-CA）31に証明書の有効性を問い合わせる。さらに、RSA認証局（RSA-CA）31は証明書の有効性を確認してECC認証局（ECC-CA）21に結果を返す。次に、ECC認証局（ECC-CA）21はECC登録局（ECC-RA）22を通してECCデバイス23に結果を返す。ECCデバイス23は受け取った証明書の有効性を確認してRSAデバイス33の認証を行う。

【0024】

このように、異なる署名方式の公開鍵証明書を保有するエンドエンティティ（EE）同士では、直接相互認証ができないため、データ通信において上述のような処理が必要となる。

【0025】

本発明は、このような公開鍵証明書を用いた公開鍵暗号方式によるデータ通信システムにおける問題点を解決することを目的とするものであり、複数の暗号化アルゴリズムをサポートし、複数の暗号化アルゴリズムに対応した公開鍵証明書を発行することにより、ECCデバイス、RSAデバイス等、特定の暗号化アルゴリズムの適用のみ可能なデバイスにおいて利用可能な複数の署名方式による署名を付加した公開鍵証明書を発行することで異なる署名アルゴリズムを処理する

デバイス相互間において公開鍵証明書の有効利用を可能とした公開鍵証明書発行システム、公開鍵証明書発行方法、および情報処理装置、情報記録媒体、並びにプログラム記憶媒体を提供することを目的とする。

【 0 0 2 6 】

【課題を解決するための手段】

本発明の第 1 の側面は、

公開鍵証明書を利用するエンティティの公開鍵証明書を発行する認証局と、

管轄エンティティから受領する公開鍵証明書発行要求を前記認証局に対して送信する登録局とを有し、

前記認証局は、各々が異なる署名方式を実行する複数の認証局によって構成され、前記登録局からの公開鍵証明書発行要求に応じて、前記複数の認証局間で公開鍵証明書を転送し、各認証局において異なる署名アルゴリズムに従って公開鍵証明書を構成するメッセージデータに対する電子署名を実行し、異なる署名アルゴリズムに従った複数の署名を格納した複数署名付き公開鍵証明書を発行する構成を有することを特徴とする公開鍵証明書発行システムにある。

【 0 0 2 7 】

さらに、本発明の公開鍵証明書発行システムの一実施態様において、前記複数の認証局は、RSA署名アルゴリズムに従った署名生成処理を実行するRSA認証局と、楕円曲線暗号（ECC）署名アルゴリズムに従った署名生成処理を実行するECC認証局とを含み、前記複数署名付き公開鍵証明書に格納された署名は、RSA署名アルゴリズムに従った署名と、楕円曲線暗号（ECC）署名アルゴリズムに従った署名とを含むことを特徴とする。

【 0 0 2 8 】

さらに、本発明の公開鍵証明書発行システムの一実施態様において、前記複数の認証局の少なくとも 1 以上の認証局は、公開鍵証明書の拡張領域に生成した署名、および生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行する構成を有することを特徴とする。

【 0 0 2 9 】

さらに、本発明の公開鍵証明書発行システムの一実施態様において、前記複数

の認証局の少なくとも1以上の認証局は、公開鍵証明書の基本領域および拡張領域以外の領域に生成した署名を格納し、拡張領域に生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行する構成を有することを特徴とする。

## 【0030】

さらに、本発明の公開鍵証明書発行システムの一実施態様において、前記複数の認証局の少なくとも1以上の認証局は、公開鍵証明書に2以上の署名が含まれるか否かを示すフラグ情報を公開鍵証明書に格納する処理を実行する構成を有することを特徴とする。

## 【0031】

さらに、本発明の第2の側面は、

公開鍵証明書を利用するエンティティの公開鍵証明書を発行する認証局と、管轄エンティティから受領する公開鍵証明書発行要求を前記認証局に対して送信する登録局とを有し、登録局からの要求に応じて公開鍵証明書を発行する公開鍵証明書発行方法において、

前記認証局は、各々が異なる署名方式を実行する複数の認証局によって構成され、

前記登録局からの公開鍵証明書発行要求に応じて、前記複数の認証局間で公開鍵証明書を転送し、各認証局において異なる署名アルゴリズムに従って公開鍵証明書を構成するメッセージデータに対する電子署名を実行し、異なる署名アルゴリズムに従った複数の署名を格納した複数署名付き公開鍵証明書を発行することを特徴とする公開鍵証明書発行方法にある。

## 【0032】

さらに、本発明の公開鍵証明書発行方法の一実施態様において、前記複数の認証局の少なくとも1以上の認証局は、署名の付加された公開鍵証明書に対して、付加された署名と異なる署名アルゴリズムを適用して署名を生成して前記公開鍵証明書に付加する署名ステップを実行することを特徴とする。

## 【0033】

さらに、本発明の公開鍵証明書発行方法の一実施態様において、前記複数の認



証局は、R S A 署名アルゴリズムに従った署名生成処理を実行する R S A 認証局と、楕円曲線暗号（E C C）署名アルゴリズムに従った署名生成処理を実行する E C C 認証局とを含み、R S A 認証局において、R S A 署名アルゴリズムに従った署名生成処理を実行し、E C C 認証局において、E C C 署名アルゴリズムに従った署名生成処理を実行し、R S A 署名アルゴリズムに従った署名と、楕円曲線暗号（E C C）署名アルゴリズムに従った署名とを含む複数署名付き公開鍵証明書を発行することを特徴とする。

## 【 0 0 3 4 】

さらに、本発明の公開鍵証明書発行方法の一実施態様において、前記複数の認証局の少なくとも 1 以上の認証局は、公開鍵証明書の拡張領域に生成した署名、および生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行することを特徴とする。

## 【 0 0 3 5 】

さらに、本発明の公開鍵証明書発行方法の一実施態様において、前記複数の認証局の少なくとも 1 以上の認証局は、公開鍵証明書の基本領域および拡張領域以外の領域に生成した署名を格納し、拡張領域に生成した署名に関する署名アルゴリズム情報を含む署名情報を格納する処理を実行することを特徴とする。

## 【 0 0 3 6 】

さらに、本発明の公開鍵証明書発行方法の一実施態様において、前記複数の認証局の少なくとも 1 以上の認証局は、公開鍵証明書に 2 以上の署名が含まれるか否かを示すフラグ情報を公開鍵証明書に格納する処理を実行することを特徴とする。

## 【 0 0 3 7 】

さらに、本発明の第 3 の側面は、

公開鍵証明書の検証処理を実行する情報処理装置であり、

公開鍵証明書内の基本領域および拡張領域に格納された署名情報に記録された複数の署名アルゴリズム中から、自己の情報処理装置において検証処理可能な署名アルゴリズムを選択して、該選択された署名アルゴリズムに従った署名の検証処理を実行する構成を有することを特徴とする情報処理装置にある。

【 0 0 3 8 】

さらに、本発明の第 4 の側面は、  
公開鍵証明書を検証処理を実行する情報処理装置であり、  
複数の署名アルゴリズムに従った署名検証処理機能を備えたことを特徴とする  
情報処理装置にある。

【 0 0 3 9 】

さらに、本発明の情報処理装置の一実施態様において、前記複数の署名アルゴリズムは、RSA 署名アルゴリズムと、楕円曲線暗号 (ECC) 署名アルゴリズムとを含むことを特徴とする。

【 0 0 4 0 】

さらに、本発明の第 5 の側面は、  
公開鍵を格納した公開鍵証明書を格納した情報記録媒体であり、  
前記公開鍵証明書は、複数の署名アルゴリズムに従った署名を格納した構成であることを特徴とする情報記録媒体にある。

【 0 0 4 1 】

さらに、本発明の情報記録媒体の一実施態様において、前記複数の署名アルゴリズムは、RSA 署名アルゴリズムと、楕円曲線暗号 (ECC) 署名アルゴリズムとを含むことを特徴とする。

【 0 0 4 2 】

さらに、本発明の第 6 の側面は、  
公開鍵証明書を利用するエンティティの公開鍵証明書を発行する公開鍵証明書発行処理をコンピュータ・システム上で実行せしめるコンピュータ・プログラムを提供するプログラム記憶媒体であって、前記コンピュータ・プログラムは、  
署名の付加された公開鍵証明書に対して、付加された署名と異なる署名アルゴリズムを適用して第 2 の署名を生成して付加する署名ステップ  
を有することを特徴とするプログラム記憶媒体にある。

【 0 0 4 3 】

なお、本発明の第 6 の側面に係るプログラム記憶媒体は、例えば、様々なプログラム・コードを実行可能な汎用コンピュータ・システムに対して、コンピュー

タ・プログラムをコンピュータ可読な形式で提供する媒体である。媒体は、CDやFD、MOなどの記録媒体、あるいは、ネットワークなどの伝送媒体など、その形態は特に限定されない。

【0044】

このようなプログラム記憶媒体は、コンピュータ・システム上で所定のコンピュータ・プログラムの機能を実現するための、コンピュータ・プログラムと提供媒体との構造上又は機能上の協働的關係を定義したものである。換言すれば、該提供媒体を介してコンピュータ・プログラムをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の他の側面と同様の作用効果を得ることができるのである。

【0045】

本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【0046】

【発明の実施の形態】

以下、図面を参照しながら、本発明の実施の形態について詳細に説明する。まず、以下の実施例中で使用する用語の意味について説明する。

認証局 (CA : Certificate Authority)

公開鍵証明書を作成、発行する機関。

登録局 (RA : Registration Authority)

公開鍵証明書を発行するための登録業務を行う。

ユーザ、サービスプロバイダ、サーバなど公開鍵証明書の利用主体であるエンドエンティティ (EE) からの公開鍵証明書発行依頼を受け、認証局 (CA) に公開鍵証明書発行要求を行う。発行された証明書はエンドエンティティ (EE) に渡される。

ハードウェア・セキュリティ・モジュール (HSM : Hardware Security Module)

署名鍵を保持し、証明書に署名付けを行う専用ハードウェア。

エンドエンティティ (EE : End Entity)

公開鍵証明書発行対象。すなわち公開鍵証明書を利用する機器やサーバ、あるいはユーザ、サービスプロバイダなどである。

【0047】

〔本発明のシステム構成の概要〕

まず、本発明の構成における公開鍵証明書発行処理、署名処理、利用処理の概要について図を用いて説明する。図4において、RSAデバイス41はRSA署名アルゴリズムの処理（検証）が実行可能なデバイスであり、ECCデバイス42はECC署名アルゴリズムの処理（検証）が実行可能なデバイスである。

【0048】

RSA認証局（RSA-CA）43はRSA署名アルゴリズムに従った認証局署名を公開鍵証明書に付加して公開鍵証明書を発行する処理を実行する認証局であり、ECC認証局（ECC-CA）44はECC署名アルゴリズムに従った認証局署名を公開鍵証明書に付加して公開鍵証明書を発行する処理を実行する認証局である。

【0049】

RSA&ECC登録局（RSA&ECC-RA）45は、各エンドエンティティ（デバイス）からの公開鍵証明書発行要求を受け付けて、RSA署名アルゴリズムとECC署名アルゴリズムに従った2つの認証局署名を付加した公開鍵証明書をRSA認証局（RSA-CA）43およびECC認証局（ECC-CA）44に依頼して作成し、各エンドエンティティに送信する処理を実行する。

【0050】

この結果、各エンドエンティティ、すなわち、RSAデバイス41、ECCデバイス42の各々は、RSA&ECC登録局（RSA&ECC-RA）45経由で発行されたRSA署名アルゴリズムとECC署名アルゴリズムに従った2つの認証局署名を付加した公開鍵証明書を有することになる。

【0051】

RSAデバイス41とECCデバイス42との双方は、公開鍵暗号方式に従った相互認証処理のために、各デバイスの公開鍵証明書を相手方に送信する。各デバイスは受信した通信相手の公開鍵証明書の複数署名中から自己のデバイスで健

勝可能なアルゴリズムに従った署名を選択して検証を実行して公開鍵証明書の正当性を確認した後、公開鍵証明書中から相手方の公開鍵を取り出して相互認証の手続きを実行する。

## 【 0 0 5 2 】

この署名検証処理に際して、RSAデバイス41は、複数署名付き公開鍵証明書中からRSA署名を取り出してRSA署名アルゴリズムに従った検証処理を実行し、ECCデバイス42は、複数署名付き公開鍵証明書中からECC署名を取り出してECC署名アルゴリズムに従った検証処理を実行して、各々が受領した公開鍵証明書の正当性検証を実行することができる。

## 【 0 0 5 3 】

本発明のシステムでは、このように異なる署名アルゴリズムの処理を実行するデバイス相互間でそれぞれの公開鍵証明書を相互に送付して署名検証が可能になる。

## 【 0 0 5 4 】

## [公開鍵証明書]

本発明で適用可能な公開鍵証明書の詳細について説明する。公開鍵証明書は、公開鍵を用いた暗号データの送受信、あるいはデータ送受信を行なう2者間での相互認証等の処理において、使用する公開鍵が正当な利用者の有する公開鍵であることを第三者、すなわち公開鍵証明書の発行局としての認証局 (CA: Certificate Authority) が証明したものである。本発明のシステムで使用される公開鍵証明書の詳細構成について図5、図6を用いて説明する。図5、6の公開鍵証明書のフォーマット例は、公開鍵証明書フォーマットX.509 V3に準拠した例である。

## 【 0 0 5 5 】

バージョン (version) は、証明書フォーマットのバージョンを示す。

シリアルナンバ (Serial Number) は、認証局 (CA) によって設定される公開鍵証明書のシリアルナンバである。

署名アルゴリズム識別子、アルゴリズムパラメータ (Signature algorithm Identifier algorithm parameter) は、公開鍵証明書の署名アルゴリズムとそのパ

ラメータを記録するフィールドである。なお、署名アルゴリズムとしては、楕円曲線暗号（ECC）、RSAなどがあり、楕円曲線暗号が適用されている場合はパラメータおよび鍵長が記録され、RSAが適用されている場合には鍵長が記録される。さらに、他の暗号方式が適用されればその方式の識別子が記録される。

発行者（issuer）は、公開鍵証明書の発行者、すなわち認証局（CA）の名称が識別可能な形式（Distinguished Name）で記録されるフィールドである。

有効期限（validity）は、証明書の有効期限である開始日時、終了日時が記録される。

サブジェクト（subject）は、ユーザである認証対象者の名前が記録される。具体的には例えばユーザ機器のIDや、サービス提供主体のID等である。

サブジェクト公開鍵情報（subject Public Key Info algorithm subject Public key）は、ユーザの公開鍵情報としての鍵アルゴリズム、鍵情報そのものを格納するフィールドである。

#### 【 0 0 5 6 】

ここまでが、公開鍵証明書フォーマットX. 509 V1に含まれるフィールドであり、以下は、公開鍵証明書フォーマットX. 509 V3において追加されるフィールドである。

#### 【 0 0 5 7 】

証明局鍵識別子（authority Key Identifier—key Identifier、authority Cert Issuer、authority Cert Serial Number）は、認証局（CA）の鍵を識別する情報であり、鍵識別番号（8進数）、認証局（CA）の名称、認証番号を格納する。

サブジェクト鍵識別子（subject key Identifier）は、複数の鍵を公開鍵証明書において証明する場合に各鍵を識別するための識別子を格納する。

鍵使用目的（key usage）は、鍵の使用目的を指定するフィールドであり、（0）デジタル署名用、（1）否認防止用、（2）鍵の暗号化用、（3）メッセージの暗号化用、（4）共通鍵配送用、（5）認証の署名確認用、（6）失効リストの署名確認用の各使用目的が設定される。

秘密鍵有効期限（private Key Usage Period）は、ユーザの有する秘密鍵の有効

期限を記録する。

認証局ポリシー (certificate Policies) は、認証局 (CA) および登録局 (RA) の証明書発行ポリシーを記録する。例えば ISO/IEC 9384-1 に準拠したポリシー ID、認証基準である。

ポリシー・マッピング (policy Mapping) は、認証局 (CA) を認証する場合にのみ記録するフィールドであり、証明書発行を行なう認証局 (CA) のポリシーと、被認証局ポリシーのマッピングを規定する。

サポート・アルゴリズム (supported Algorithms) は、ディレクトリ (X. 500) のアトリビュートを定義する。これは、コミュニケーションの相手がディレクトリ情報を利用する場合に、事前にそのアトリビュートを知らせるのに用いる。

サブジェクト別名 (subject Alt Name) は、ユーザの別名を記録するフィールドである。

発行者別名 (issuer Alt Name) は、証明書発行者の別名を記録するフィールドである。

サブジェクト・ディレクトリ・アトリビュート (subject Directory Attribute) は、ユーザの任意の属性を記録するフィールドである。本発明の構成では、本フィールドに第2の署名に関する署名アルゴリズム、パラメータ情報、および第2の署名を格納する場合がある。

基本制約 (basic Constraint) は、証明対象の公開鍵が認証局 (CA) の署名用か、ユーザのものかを区別するためのフィールドである。

許容サブトリー制約名 (name Constraints permitted Subtrees) は、被認証者が認証局 (CA) である場合にのみ使用される証明書の有効領域を示すフィールドである。

制約ポリシー (policy Constraints) は、認証パスの残りに対する明確な認証ポリシー ID、禁止ポリシーマップを要求する制限を記述する。

CRL 参照ポイント (Certificate Revocation List Distribution Points) は、ユーザが証明書を利用する際に、証明書が失効していないか、どうかを確認するための失効リストの参照ポイントを記述するフィールドである。

署名は、認証局（CA）の署名フィールドである。

【0058】

なお、本発明の構成においては、署名は、単一のものばかりではなく、異なる署名アルゴリズムによる複数の署名が公開鍵証明書に付加される。この複数署名構成については、後段で説明する。

【0059】

[署名アルゴリズム]

上述の公開鍵証明書の署名は、公開鍵証明書発行局（CA）の秘密鍵を用いて公開鍵証明書のデータに対して実行される電子署名であり、公開鍵証明書の利用者は、公開鍵証明書発行局（CA）の公開鍵を用いて検証を行ない、公開鍵証明書の正当性、改竄有無がチェック可能となっている。

【0060】

電子署名のアルゴリズムについて、まず、図7を用いて楕円曲線暗号（Elliptic Curve Cryptography（ECC））を用いた処理について説明する。図7に示す処理は、ECDSA（（Elliptic Curve Digital Signature Algorithm）、IEEE P1363/D3）を用いた電子署名データの生成処理フローである。

【0061】

図7の各ステップについて説明する。ステップS1において、 $p$ を標数、 $a$ 、 $b$ を楕円曲線の係数（楕円曲線： $4a^3 + 27b^2 \neq 0 \pmod{p}$ ）、 $G$ を楕円曲線上のベースポイント、 $r$ を $G$ の位数、 $K_s$ を秘密鍵（ $0 < K_s < r$ ）とする。ステップS2において、メッセージ $M$ のハッシュ値を計算し、 $f = \text{Hash}(M)$ とする。

【0062】

ここで、ハッシュ関数を用いてハッシュ値を求める方法を説明する。ハッシュ関数とは、メッセージを入力とし、これを所定のビット長のデータに圧縮し、ハッシュ値として出力する関数である。ハッシュ関数は、ハッシュ値（出力）から入力を予測することが難しく、ハッシュ関数に入力されたデータの1ビットが変化したとき、ハッシュ値の多くのビットが変化し、また、同一のハッシュ値を持つ異なる入力データを探し出すことが困難である特徴を有する。ハッシュ関数としては、MD4、MD5、SHA-1などが用いられる場合もあるし、DES-



CBCが用いられる場合もある。この場合は、最終出力値となるMAC（チェック値：ICVに相当する）がハッシュ値となる。

## 【0063】

続けて、ステップS3で、乱数 $u$  ( $0 < u < r$ ) を生成し、ステップS4でベースポイントを $u$ 倍した座標 $V(X_v, Y_v)$ を計算する。なお、楕円曲線上の加算、2倍算は次のように定義されている。

## 【0064】

## 【数1】

$P=(X_a, Y_a), Q=(X_b, Y_b), R=(X_c, Y_c)=P+Q$ とすると、

$P \neq Q$ の時（加算）、

$$X_c = \lambda^2 - X_a - X_b$$

$$Y_c = \lambda \times (X_a - X_c) - Y_a$$

$$\lambda = (Y_b - Y_a) / (X_b - X_a)$$

$P=Q$ の時（2倍算）、

$$X_c = \lambda^2 - 2X_a$$

$$Y_c = \lambda \times (X_a - X_c) - Y_a$$

$$\lambda = (3(X_a)^2 + a) / (2Y_a)$$

## 【0065】

これらを用いて点 $G$ の $u$ 倍を計算する（速度は遅いが、最もわかりやすい演算方法として次のように行う。 $G, 2 \times G, 4 \times G \dots$ を計算し、 $u$ を2進数展開して1が立っているところに対応する $2^i \times G$  ( $G$ を $i$ 回2倍算した値 ( $i$ は $u$ のLSBから数えた時のビット位置))を加算する。

## 【0066】

ステップS5で、 $c = X_v \bmod r$ を計算し、ステップS6でこの値が0になるかどうか判定し、0でなければステップS7で $d = [(f + cK_s) / u] \bmod r$ を計算し、ステップS8で $d$ が0であるかどうか判定し、 $d$ が0でなければ、ステップS9で $c$ および $d$ を電子署名データとして出力する。仮に、 $r$ を160ビット長の長さであると仮定すると、電子署名データは320ビット長となる。

## 【0067】

ステップS6において、 $c$ が0であった場合、ステップS3に戻って新たな乱数を生成し直す。同様に、ステップS8で $d$ が0であった場合も、ステップS3に戻って乱数を生成し直す。

## 【0068】

次に、ECCによる電子署名の検証方法を、図8を用いて説明する。ステップS11で、 $M$ をメッセージ、 $p$ を標数、 $a$ 、 $b$ を楕円曲線の係数（楕円曲線： $y^2 = x^3 + ax + b$ ）、 $G$ を楕円曲線上のベースポイント、 $r$ を $G$ の位数、 $G$ および $Ks \times G$ を公開鍵（ $0 < Ks < r$ ）とする。ステップS12で電子署名データ $c$ および $d$ が $0 < c < r$ 、 $0 < d < r$ を満たすか検証する。これを満たしていた場合、ステップS13で、メッセージ $M$ のハッシュ値を計算し、 $f = \text{Hash}(M)$ とする。次に、ステップS14で $h = 1/d \bmod r$ を計算し、ステップS15で $h1 = fh \bmod r$ 、 $h2 = ch \bmod r$ を計算する。

## 【0069】

ステップS16において、既に計算した $h1$ および $h2$ を用い、点 $P = (Xp, Yp) = h1 \times G + h2 \cdot Ks \times G$ を計算する。電子署名検証者は、公開鍵 $G$ および $Ks \times G$ を知っているため、図7のステップS4と同様に楕円曲線上の点のスカラー倍の計算ができる。そして、ステップS17で点 $P$ が無限遠点かどうか判定し、無限遠点でなければステップS18に進む（実際には、無限遠点の判定はステップS16でできてしまう。つまり、 $P = (X, Y)$ 、 $Q = (X, -Y)$ の加算を行うと、 $\lambda$ が計算できず、 $P + Q$ が無限遠点であることが判明している）。ステップS18で $Xp \bmod r$ を計算し、電子署名データ $c$ と比較する。最後に、この値が一致していた場合、ステップS19に進み、電子署名が正しいと判定する。

## 【0070】

電子署名が正しいと判定された場合、データは改竄されておらず、公開鍵に対応した秘密鍵を保持する者が電子署名を生成したことがわかる。

## 【0071】

ステップS12において、電子署名データ $c$ または $d$ が、 $0 < c < r$ 、 $0 < d$

$< r$  を満たさなかった場合、ステップ S20 に進む。また、ステップ S17 において、点 P が無限遠点であった場合もステップ S20 に進む。さらにまた、ステップ S18 において、 $X_p \bmod r$  の値が、電子署名データ c と一致していなかった場合にもステップ S20 に進む。

## 【0072】

ステップ S20 において、電子署名が正しくないと判定された場合、データは改竄されているか、公開鍵に対応した秘密鍵を保持する者が電子署名を生成したのではないことがわかる。

## 【0073】

次に、RSA 暗号方式の署名アルゴリズムを図 9、図 10 を用いて説明する。図 9 は、RSA 暗号方式の署名および署名検証用の公開鍵、秘密鍵の生成方法を示し、図 10 は (a) 署名生成処理、(b) 署名検証処理を示している。

## 【0074】

図 9 の署名および署名検証用の公開鍵、秘密鍵の生成フローにおいて、まず素数  $p$ 、 $q$  (150 桁程度) を選択し (S21)、 $n = pq$  を計算し (S22)、さらに、 $L = (p-1)(q-1)$  を計算し (S23)、 $L$  と共通因数を持たない  $n$  未満の正整数  $e$  を選択し、 $(n, e)$  を公開鍵とし (S24)、 $de = 1 \bmod L$  を満足する正整数  $d$  を求めて  $(p, q, d)$  を秘密鍵とする (S25)。

## 【0075】

このような公開鍵、秘密鍵を用いた署名の生成、検証は図 10 のフローに従って行われる。図 10 (a) に示す署名生成は、ステップ S31 において、署名対象となるメッセージ  $M$  に対してハッシュ関数  $h$  を適用して  $m = h(M)$  を生成し、さらに、 $S = m^d \bmod n$  を生成 (S32) して、 $S$  を署名とする。

## 【0076】

図 10 (b) に示す署名検証は、署名検証対象となるメッセージ  $M$  に対してハッシュ関数  $h$  を適用して  $m = h(M)$  を生成 (S33) し、さらに、 $m = S^e \bmod n$  が成立するか否かを検証 (S34) して、成立する場合は、署名が正しい (S35) と判定する。

## 【0077】

署名が正しいと判定された場合、データは改竄されておらず、公開鍵に対応した秘密鍵を保持する者が電子署名を生成したことがわかる。

## 【0078】

ステップS34において、 $m = S^e \bmod n$ が成立しないと判定された場合、ステップS36において署名が間違っていると判定され、データは改竄されているか、公開鍵に対応した秘密鍵を保持する者が電子署名を生成したのではないことがわかる。

## 【0079】

このように、公開鍵証明書の署名検証により、公開鍵証明所の正当性が検証されることになる。上述したように、署名の検証は、署名アルゴリズムに従った暗号処理を実行することが必要であり、エンドエンティティの機器において実行可能な署名アルゴリズムであることが必要であり、一般には登録局(RA)の管理の下に共通の署名アルゴリズムが用いられる。

## 【0080】

## [署名態様]

図11に一般的な公開鍵証明書の署名処理態様を説明する図を示す。前述したように公開鍵証明書は、基本データからなる基本領域(V1領域)61、拡張データからなる拡張領域(V3領域)62を持つ。署名は、これら基本領域(V1領域)61、拡張領域(V3領域)62全体に基づいて得られるデータをメッセージとして、前述の楕円曲線暗号(ECC)、あるいはRSA署名生成方法などのアルゴリズムを適用して生成する。署名生成処理の結果は署名領域63に記録される。なお、署名生成アルゴリズム、パラメータなどの情報は、前述したように基本領域(V1領域)61の署名情報領域64に記録される。

## 【0081】

本発明の公開鍵証明書の署名処理態様を説明する図を図12および図13に示す。まず、図12の構成について説明する。図12の公開鍵証明書は、前述と同様、基本データからなる基本領域(V1領域)71、拡張データからなる拡張領域(V3領域)72を持つ。署名領域A、73は、図11の構成と同様の署名で

あり、基本領域（V1領域）71、拡張領域（V3領域）72全体に基づいて得られるデータをメッセージとして、前述の楕円曲線暗号（ECC）、あるいはRSA署名生成方法などのアルゴリズムを適用して生成された署名結果を格納する。署名領域A、73に格納される署名に関する署名生成アルゴリズム、パラメータなどの情報は、基本領域（V1領域）71の署名情報領域74に記録される。

## 【0082】

さらに、署名領域B、75は、署名領域A、73に格納される署名とは異なる署名アルゴリズムを適用して、基本領域（V1領域）71、拡張領域（V3領域）72全体に基づいて得られるデータをメッセージとして生成された署名である。署名領域B、75は拡張領域（V3領域）72に組み込まれて格納される。例えば拡張領域（V3領域）72のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）内に格納可能である。あるいはその他のフィールドのデータ格納可能な領域に格納してもよい。

## 【0083】

さらに、拡張領域（V3領域）72には、通常の署名（署名領域A、73の署名）と異なるアルゴリズムによる署名が含まれるか否かを示すフラグ情報76と、署名領域B、75に格納される署名に関する署名生成アルゴリズム、パラメータなどの情報が署名情報領域77に記録される。署名情報領域77は、例えば拡張領域（V3領域）72のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）内に形成することができる。あるいはその他のフィールドのデータ格納可能な領域を用いてもよい。

## 【0084】

フラグ情報76は、例えばフラグ=1のときには、署名領域B、75に署名領域A、73に格納される署名とは異なる署名アルゴリズムを適用して生成された署名が格納されていることを示し、フラグ=0のときには、署名領域B、75に署名が格納されていないことを示す。

## 【0085】

具体的には、例えば、署名領域A、73の署名がECCアルゴリズムに従った署名である場合、署名領域B、75の署名がRSAアルゴリズムに従った署名と

なり、署名領域 A， 7 3 の署名が R S A アルゴリズムに従った署名である場合、署名領域 B， 7 5 の署名が E C C アルゴリズムに従った署名となる。

#### 【 0 0 8 6 】

図 1 2 の例では、拡張領域（V 3 領域） 7 2 に署名領域 A， 7 3 の署名アルゴリズムと異なるアルゴリズムによる第 2 の署名を 1 つ格納した例を示しているが、さらに異なるアルゴリズム、あるいは異なる鍵長、パラメータを適用した署名を生成して拡張領域（V 3 領域） 7 2 に複数の署名を格納してもよい。この場合は、例えばフラグを 2 ビット構成として 0 0 の場合は、第 2 の署名なし、0 1 の場合は、第 2 の署名あり、1 0 の場合は、第 3 の署名もある等、複数の署名が含まれることを識別可能な構成とする。さらに署名情報領域 7 7 についても格納する署名の数に対応する数の情報を格納する。

#### 【 0 0 8 7 】

このように複数のアルゴリズムに従った署名を持つ公開鍵証明書の署名検証は、例えば、署名領域 A， 7 3 の署名が E C C アルゴリズムに従った署名であり、署名領域 B， 7 5 の署名が R S A アルゴリズムに従った署名である場合、E C C アルゴリズムによる検証のみが可能なエンドエンティティ（E E）のデバイスは、署名領域 A， 7 3 の署名検証を実行し、R S A アルゴリズムによる検証のみが可能なエンドエンティティ（E E）のデバイスは、署名領域 B， 7 5 の署名検証を実行することが可能となり、E C C デバイスと R S A デバイスとの間の相互認証処理が実行可能となる。

#### 【 0 0 8 8 】

次に、図 1 3 の構成について説明する。図 1 3 の公開鍵証明書は、前述と同様、基本データからなる基本領域（V 1 領域） 8 1、拡張データからなる拡張領域（V 3 領域） 8 2 を持つ。署名領域 A， 8 3 は、図 1 2 の構成と同様の署名であり、基本領域（V 1 領域） 8 1、拡張領域（V 3 領域） 8 2 全体に基づいて得られるデータをメッセージとして、前述の楕円曲線暗号（E C C）、あるいは R S A 署名生成方法などのアルゴリズムを適用して生成された署名結果を格納する。署名領域 A， 8 3 に格納される署名に関する署名生成アルゴリズム、パラメータなどの情報は、基本領域（V 1 領域） 8 1 の署名情報領域 8 4 に記録される。

## 【0089】

さらに、署名領域B, 85は、署名領域A, 83に格納される署名とは異なる署名アルゴリズムを適用して、基本領域(V1領域)81、拡張領域(V3領域)82全体に基づいて得られるデータをメッセージとして生成された署名である。図13の構成では、署名領域B, 85は拡張領域(V3領域)の外に格納される。拡張領域(V3領域)82には、通常の署名(署名領域A, 83の署名)と異なるアルゴリズムによる署名が含まれるか否かを示すフラグ情報86と、署名領域B, 85に格納される署名に関する署名生成アルゴリズム、パラメータなどの情報が署名情報領域87に記録される。フラグ情報の意味は、図12の場合と同様である。

## 【0090】

なお、図13の例でも図12と同様、署名領域A, 83の署名アルゴリズムと異なるアルゴリズムによる第2の署名を、署名領域B, 85に1つ格納した例を示しているが、さらに異なるアルゴリズム、あるいは異なる鍵長、パラメータを適用した署名を生成して格納してもよい。署名領域C、Dを設けてもよい。

## 【0091】

図13の構成においても、図12の構成と同様、複数のアルゴリズムに従った署名を持つ公開鍵証明書が生成されることになり、それぞれが異なるアルゴリズムによる検証のみが可能なエンドエンティティ(EE)のデバイス相互間で、署名検証を実行することが可能となり、ECCデバイスとRSAデバイスとの間の相互認証処理が実行可能となる。

## 【0092】

## [公開鍵証明書発行、署名生成処理]

次に、公開鍵証明書発行、署名生成処理について説明する。ここでは、図12または図13を用いて説明した複数の署名アルゴリズムに従った複数の署名を持つ公開鍵証明書の発行処理について説明する。

## 【0093】

まず、図14に示すフローチャートに従って、図12に示す複数署名付き公開鍵証明書の発行処理、署名処理について説明する。

## 【0094】

まず、公開鍵証明書の発行要求デバイスは、登録局（RA）に対して公開鍵証明書の発行要求を送信（S301）する。なお、登録局（RA）は、RSA署名アルゴリズムおよびECC署名アルゴリズムの2つのアルゴリズムの署名を付加した公開鍵証明書の発行要求を受け付ける登録局（RSA&ECC-RA）である。

## 【0095】

登録局（RSA&ECC-RA）が公開鍵証明書の発行要求を受信すると、発行要求デバイスの確認、登録など必要な処理を実行後、公開鍵証明書の基本領域にセットする署名アルゴリズム（この場合はRSAとする）に対応する認証局（RSA-CA）に証明書の発行要求を送信（S302）する。なお、本フローの処理例では、まず第1の署名をRSAとし、第2の署名をECCアルゴリズムに従った形式として説明するが、この逆の形態も同様に実行可能である。

## 【0096】

登録局から公開鍵証明書の発行要求を受信した認証局（RSA-CA）は、公開鍵証明書の生成を実行する。この際、公開鍵証明書の基本領域に署名アルゴリズムとしてRSAアルゴリズム、およびパラメータをセットする。さらに、認証局（RSA-CA）は、他の方式の署名を実行する認証局（ECC-CA）に生成した公開鍵証明書を送信（S303）する。

## 【0097】

認証局（RSA-CA）から公開鍵証明書データを受信した認証局（ECC-CA）は、公開鍵証明書の受信した公開鍵証明書の拡張領域のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）内に署名アルゴリズムとしてECCアルゴリズム、およびパラメータを署名情報としてセットし、さらに、第2の署名が存在することを示すフラグをセット（S304）する。なお、署名情報はサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）以外のフィールドのデータ格納可能な領域に格納してもよい。

## 【0098】

次に、認証局（ECC-CA）は、公開鍵証明書のデータ（メッセージ）に基



づいて ECC アルゴリズムに従った署名生成処理を実行する。生成した署名は、例えば拡張領域のサブジェクト・ディレクトリ・アトリビュート (subject Directory Attribute) に格納する。署名生成、格納が終了すると、認証局 (ECC-CA) は、公開鍵証明書を送信局 (RSA-CA) に送信 (S305) する。

## 【0099】

認証局 (ECC-CA) から公開鍵証明書データを受信した認証局 (RSA-CA) は、公開鍵証明書のデータ (メッセージ) に基づいて RSA アルゴリズムに従った署名生成処理を実行して公開鍵証明書の署名フィールドに格納し、登録局に送信 (S306) する。

## 【0100】

登録局は、受信した公開鍵証明書をデバイスに送信 (S307) し、デバイスは受信した公開鍵証明書を記憶手段に格納 (S308) する。

## 【0101】

次に、図 15 に示すフローチャートに従って、図 13 に示す複数署名付き公開鍵証明書の発行処理、署名処理について説明する。

## 【0102】

まず、公開鍵証明書の発行要求デバイスは、登録局 (RA) に対して公開鍵証明書の発行要求を送信 (S351) する。なお、登録局 (RA) は、RSA 署名アルゴリズムおよび ECC 署名アルゴリズムの 2 つのアルゴリズムの署名を付加した公開鍵証明書の発行要求を受け付ける登録局 (RSA & ECC-RA) である。

## 【0103】

登録局 (RSA & ECC-RA) が、公開鍵証明書の発行要求を受信すると、発行要求デバイスの確認、登録など必要な処理を実行後、公開鍵証明書の基本領域にセットする署名アルゴリズム (この場合は RSA とする) に対応する認証局 (RSA-CA) に証明書の発行要求を送信 (S352) する。なお、本フローの処理例においても、まず第 1 の署名を RSA とし、第 2 の署名を ECC アルゴリズムに従った形式として説明するが、この逆の形態も同様に実行可能である。

## 【0104】

登録局から公開鍵証明書の発行要求を受信した認証局（RSA-CA）は、公開鍵証明書の生成を実行する。この際、公開鍵証明書の基本領域に署名アルゴリズムとしてRSAアルゴリズム、およびパラメータをセットする。さらに、認証局（RSA-CA）は、他の方式の署名を実行する認証局（ECC-CA）に生成した公開鍵証明書を送信（S353）する。

## 【0105】

認証局（RSA-CA）から公開鍵証明書データを受信した認証局（ECC-CA）は、公開鍵証明書の受信した公開鍵証明書の拡張領域のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）内に署名アルゴリズムとしてECCアルゴリズム、およびパラメータを署名情報としてセットし、さらに、第2の署名が存在することを示すフラグをセット（S354）する。なお、署名情報はサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）以外のフィールドのデータ格納可能な領域に格納してもよい。

## 【0106】

次に、認証局（ECC-CA）は、公開鍵証明書のデータ（メッセージ）に基づいてECCアルゴリズムに従った署名生成処理を実行する。生成した署名は、基本領域、拡張領域外の領域に格納する。署名生成、格納が終了すると、認証局（ECC-CA）は、公開鍵証明書を認証局（RSA-CA）に送信（S355）する。

## 【0107】

認証局（ECC-CA）から公開鍵証明書データを受信した認証局（RSA-CA）は、公開鍵証明書のデータ（メッセージ）に基づいてRSAアルゴリズムに従った署名生成処理を実行して公開鍵証明書の署名フィールドに格納し、登録局に送信（S356）する。

## 【0108】

登録局は、受信した公開鍵証明書をデバイスに送信（S357）し、デバイスは受信した公開鍵証明書を記憶手段に格納（S358）する。

## 【0109】

上述したように、公開鍵証明書には複数の署名アルゴリズムに従った複数の署

名が格納され、いずれのアルゴリズムでの検証も可能となる。なお、上述の処理フローでは、2つの署名アルゴリズムのみを示したがさらに第3、4…の署名アルゴリズムに従った署名を生成して格納することも可能であり、各認証局（XX X-CA）が各CAに対応する署名アルゴリズム（XXX）で署名を実行して署名後の公開鍵証明書を他のCAに転送し、署名を順次付加すればよい。

#### 【0110】

このように、複数のアルゴリズムに従った署名を持つ公開鍵証明書が生成されることにより、ECCあるいはRSAのみの異なるアルゴリズム処理が可能な2つのデバイス間での相互認証において、相互に通信相手の公開鍵証明書の署名検証を実行することが可能となる。

#### 【0111】

##### 〔署名検証処理〕

RSA署名アルゴリズムのみの処理が可能なデバイス、ECC署名アルゴリズムのみの処理が可能なデバイスが混在する環境において、各デバイス間において相互認証、暗号化データの送受信が実行される場合の各デバイス（情報処理装置）間の通信について、処理形態を分類して以下説明する。

#### 【0112】

まず、図12の形式、すなわち拡張領域に少なくとも第2の署名が格納され、さらに第3、第4…の署名が格納された可能性もある複数署名付き公開鍵証明書を利用し、デバイスがRSAまたはECC方式などある1つのアルゴリズムの署名のみ処理可能なデバイスであるときの署名検証処理例を図16、図17の処理フローに従って説明する。

#### 【0113】

ステップS401において、デバイスは、例えば相互認証処理を実行しようとする通信相手から複数署名付き公開鍵証明書（図12参照）を受信する。公開鍵証明書を受信したデバイスは、公開鍵証明書の基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドのデータに基づいて署名アルゴリズムを確認し、自デバイスにおいて処理（検証）可能か否かを検証（S402）する。

## 【0114】

自デバイスにおいて処理（検証）可能である場合は、ステップS403において、基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドに記録された署名方式に従った検証アルゴリズムを適用して署名検証を実行する。一方、自デバイスにおいて処理（検証）可能でない場合は、ステップS404の処理を実行する。ステップS404の処理は、図17の処理フローに含まれる処理と同様であるので、図17に従って説明する。

## 【0115】

図17のステップS501で、デバイスは、例えば相互認証処理を実行しようとする通信相手から複数署名付き公開鍵証明書（図12参照）を受信する。公開鍵証明書を受信したデバイスは、公開鍵証明書の基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドのデータに基づいて署名アルゴリズムを確認し、自デバイスにおいて処理（検証）可能か否かを検証（S502）する。

## 【0116】

自デバイスにおいて処理（検証）可能である場合は、ステップS503において、基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドに記録された署名方式に従った検証アルゴリズムを適用して署名検証を実行する。一方、自デバイスにおいて処理（検証）可能でない場合は、ステップS504において、他の署名方式による署名（第2の署名）が格納されているか否かを公開鍵証明書の拡張領域のフラグによって確認する。

## 【0117】

フラグが第2の署名の格納を示していない場合（ex. フラグ=0）は、署名を検証することができずエラー（S507）となる。

## 【0118】

一方、フラグが第2の署名の格納を示している場合（ex. フラグ=1）は、第2の署名に関する署名情報を格納した例えば拡張領域のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）のデータを参照し、署名アルゴリズムが自デバイスで処理可能なアルゴリズムであるか否かを確認（

S505)する。自デバイスにおいて処理(検証)可能である場合は、ステップS506において、拡張領域のサブジェクト・ディレクトリ・アトリビュート(subject Directory Attribute)フィールドに記録された署名方式に従った検証アルゴリズムを適用して署名検証を実行する。

#### 【0119】

一方、ステップS505における第2の署名に関する署名情報の示す署名アルゴリズムが自デバイスで処理可能なアルゴリズムでない場合は、ステップS508に進み、さらに、他の署名アルゴリズムによる署名が格納されているか否かを検証する。この検証は第3の署名情報、第4の署名情報を参照する処理であり、第2の署名情報と同様、拡張領域のサブジェクト・ディレクトリ・アトリビュート(subject Directory Attribute)フィールド等の領域を参照して実行される。

#### 【0120】

さらに、異なる署名アルゴリズムによる署名が格納されている場合は、ステップS504のフラグ情報検証、ステップS505の署名検証の可否判定を実行し、署名検証可能であれば、ステップS506において署名検証を実行する。すべての署名情報を参照し、すべて検証不可と判定された場合は、ステップS508の判定において、すべての署名データについての検証可否の検証が終了となり、検証処理の実行は不可と判定され処理を終了する。

#### 【0121】

このように、特定の署名アルゴリズムに従った署名の検証のみを実行可能なデバイスであっても、複数の署名アルゴリズムに従った複数の署名を格納した公開鍵証明書を受信した場合は、格納された署名のいずれかが自デバイスにおいて検証可能であるか否かを判定し、検証可能な署名について検証処理を実行することで公開鍵証明書の正当性を判定することができる。

#### 【0122】

次に、図13の形式、すなわち公開鍵証明書の基本領域、拡張領域以外の領域に少なくとも第2の署名が格納され、さらに第3、第4…の署名が格納された可能性もある複数署名付き公開鍵証明書を利用し、デバイスがRSAまたはECC

方式などある 1 つのアルゴリズムの署名のみ処理可能なデバイスであるときの署名検証処理例を図 1 8、図 1 9 の処理フローに従って説明する。

【 0 1 2 3 】

ステップ S 6 0 1 において、デバイスは、例えば相互認証処理を実行しようとする通信相手から複数署名付き公開鍵証明書（図 1 3 参照）を受信する。公開鍵証明書を受信したデバイスは、公開鍵証明書の基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドのデータに基づいて署名アルゴリズムを確認し、自デバイスにおいて処理（検証）可能か否かを検証（S 6 0 2）する。

【 0 1 2 4 】

自デバイスにおいて処理（検証）可能である場合は、ステップ S 6 0 3 において、基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドに記録された署名方式に従った検証アルゴリズムを適用して署名検証を実行する。一方、自デバイスにおいて処理（検証）可能でない場合は、ステップ S 6 0 4 の処理を実行する。ステップ S 6 0 4 の処理は、図 1 9 に示す処理フローに含まれる処理と同様であるので、図 1 9 に従って説明する。

【 0 1 2 5 】

図 1 9 のステップ S 7 0 1 で、デバイスは、例えば相互認証処理を実行しようとする通信相手から複数署名付き公開鍵証明書（図 1 3 参照）を受信する。公開鍵証明書を受信したデバイスは、公開鍵証明書の基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドのデータに基づいて署名アルゴリズムを確認し、自デバイスにおいて処理（検証）可能か否かを検証（S 7 0 2）する。

【 0 1 2 6 】

自デバイスにおいて処理（検証）可能である場合は、ステップ S 7 0 3 において、基本領域の署名アルゴリズム識別子（Signature algorithm Identifier）フィールドに記録された署名方式に従った検証アルゴリズムを適用して署名検証を実行する。一方、自デバイスにおいて処理（検証）可能でない場合は、ステップ S 5 7 4 において、他の署名方式による署名（第 2 の署名）が格納されているか

否かを公開鍵証明書の特長領域のフラグによって確認する。

【0127】

フラグが第2の署名の格納を示していない場合（ex. フラグ=0）は、署名を検証することができずエラー（S707）となる。

【0128】

一方、フラグが第2の署名の格納を示している場合（ex. フラグ=1）は、第2の署名に関する署名情報を格納した例えば特長領域のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）のデータを参照し、署名アルゴリズムが自デバイスで処理可能なアルゴリズムであるか否かを確認（S705）する。自デバイスにおいて処理（検証）可能である場合は、ステップS706において、基本領域、特長領域外の署名フィールド（第2の署名フィールド）に格納された署名について、特長領域のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）フィールドに記録された署名方式に従った検証アルゴリズムを適用して署名検証を実行する。

【0129】

一方、ステップS705における第2の署名に関する署名情報の示す署名アルゴリズムが自デバイスで処理可能なアルゴリズムでない場合は、ステップS708に進み、さらに、他の署名アルゴリズムによる署名が格納されているか否かを検証する。この検証は第3の署名情報、第4の署名情報を参照する処理であり、第2の署名情報と同様、特長領域のサブジェクト・ディレクトリ・アトリビュート（subject Directory Attribute）フィールド等の領域を参照して署名アルゴリズムを確認して実行される。

【0130】

さらに、異なる署名アルゴリズムによる署名が格納されている場合は、ステップS704のフラグ情報検証、ステップS705の署名検証の可否判定を実行し、署名検証可能であれば、ステップS706において署名検証を実行する。すべての署名情報を参照し、すべて検証不可と判定された場合は、ステップS708の判定において、すべての署名データについての検証可否の検証が終了となり、検証処理の実行は不可と判定され処理を終了する。

## 【0131】

このように、特定の署名アルゴリズムに従った署名の検証のみを実行可能なデバイスであっても、複数の署名アルゴリズムに従った複数の署名を格納した公開鍵証明書を受信した場合は、格納された署名のいずれかが自デバイスにおいて検証可能であるか否かを判定し、検証可能な署名について検証処理を実行することで公開鍵証明書の正当性を判定することができる。

## 【0132】

【異なる署名方式を処理可能なデバイス間の相互認証処理】

次に、具体的なデバイス間の相互認証処理について図20を用いて説明する。図20において、RSAデバイス501はRSA署名アルゴリズムの処理（検証）が実行可能なデバイスであり、ECCデバイス502はECC署名アルゴリズムの処理（検証）が実行可能なデバイスである。

## 【0133】

RSA認証局（RSA-CA）503はRSA署名アルゴリズムに従った認証局署名を公開鍵証明書に付加して公開鍵証明書を発行する処理を実行する認証局であり、ECC認証局（ECC-CA）504はECC署名アルゴリズムに従った認証局署名を公開鍵証明書に付加して公開鍵証明書を発行する処理を実行する認証局である。

## 【0134】

RSA&ECC登録局（RSA&ECC-RA）505は、各エンドエンティティ（デバイス）からの公開鍵証明書発行要求を受け付けて、RSA署名アルゴリズムとECC署名アルゴリズムに従った2つの認証局署名を付加した公開鍵証明書（図12、図13参照）をRSA認証局（RSA-CA）503およびECC認証局（ECC-CA）504に依頼して作成し、各エンドエンティティに送信する処理を実行する。

## 【0135】

このような状況において、図20に示す各エンドエンティティ、すなわち、RSAデバイス501、ECCデバイス502の各々は、RSA&ECC登録局（RSA&ECC-RA）505経由で発行されたRSA署名アルゴリズムとEC



C署名アルゴリズムに従った2つの認証局署名を付加した公開鍵証明書を有する。

#### 【0136】

RSAデバイス501とECCデバイス502との双方は、公開鍵暗号方式に従った相互認証処理のために、各デバイスの公開鍵証明書を相手方に送信する。各デバイスは受信した通信相手の公開鍵証明書の署名検証を実行して公開鍵証明書の正当性を確認した後、公開鍵証明書中から相手方の公開鍵を取り出して相互認証の手続きを実行する。

#### 【0137】

この署名検証処理に際して、RSAデバイス501は、複数署名付き公開鍵証明書中からRSA署名を取り出してRSA署名アルゴリズムに従った検証処理（図10参照）を実行し、ECCデバイス502は、複数署名付き公開鍵証明書中からECC署名を取り出してECC署名アルゴリズムに従った検証処理（図8参照）を実行して、各々が受領した公開鍵証明書の正当性検証を実行することができる。

#### 【0138】

なお、RSA署名アルゴリズム、ECC署名アルゴリズム以外の第3の署名アルゴリズムを適用した場合であっても第3の署名アルゴリズムに従った署名を格納した複数署名付き公開鍵証明書を各デバイスに対して発行することによって図20と同様各デバイス間での相互認証が可能となる。

#### 【0139】

〔複数の署名モジュールを持つ認証局（CA）〕

上述の実施例において、認証局は、RSAあるいはECCなど唯一の署名アルゴリズムに従った署名を実行する構成として説明したが1つの認証局において複数の署名アルゴリズムに従って署名を可能とした認証局構成について以下、説明する。

#### 【0140】

複数の署名モジュールを持つ電子認証装置としての認証局（CA）構成について説明する。公開鍵暗号系を用いたシステムにおいて秘密鍵の保持方法や署名付

けのセキュリティの確保は認証局(CA: Certificate Authority)を構築する際の一つの課題であり、また、署名付けの演算速度向上は認証局のシステム性能の向上をもたらす。

#### 【0141】

セキュリティ確保、演算速度向上の解決策の1つとして署名鍵(秘密鍵)の保持、署名付けを専用ハードウェア(HSM: Hardware Security Module)で行うことが可能である。HSMは高い耐タンパー性を持つため、セキュリティレベル向上にも大きな役割を果たす。しかし、専用ハードウェア(HSM)で実行される暗号化アルゴリズムは固定的なものとなってしまう、署名アルゴリズムを変更して実行する運用は困難である。

#### 【0142】

本実施例のシステムにおいては、認証局(CA)は、複数の異なる署名方式、鍵長、パラメータを適用可能とした構成を持つ。具体的にはそれぞれが異なる署名アルゴリズムを実行する専用ハードウェア(HSM)、またはソフトウェアによる複数の署名モジュールを有する認証局(CA)構成を実現する。

#### 【0143】

図21に複数の署名モジュールを持つ電子認証装置としての認証局(CA)の処理を説明する図を示す。認証局(CA)700のCAサーバ701は、公開鍵証明書を利用する機器やサーバ、あるいはユーザ、サービスプロバイダなど公開鍵証明書発行対象エンドエンティティ(EE: End Entity)からの公開鍵証明書発行要求を様々な登録局(RA: Registration Authority)751~755を介して受信する。

#### 【0144】

各登録局(RA)751~755は、自己の管轄するエンドエンティティ(EE)に対して発行する公開鍵証明書に対する署名アルゴリズム、例えば、RSA(Rivest-Shamir-Adleman)暗号方式、あるいは楕円曲線暗号(ECC: Elliptic Curve Cryptography)などを許容する署名方式として規程しており、規程された署名方式から1つまたは複数のアルゴリズムによる署名を実行した公開鍵証明書の発行要求を認証局700に対して実行する。各登録局(RA)751~75

5は、自己の管轄するエンドエンティティ（EE）において処理可能、すなわち検証処理可能な暗号化アルゴリズムに従った署名がなされた公開鍵証明書の発行を要求することになり、各登録局（RA）751～755ごとに希望する署名アルゴリズムは異なることになる。

## 【0145】

公開鍵証明書の発行要求は、認証局（CA）700のCAサーバ701によって受け付けられ、CAサーバの有する各登録局（RA）751～755と適用署名アルゴリズムの種類を対応付けたテーブルに従って、署名モジュール702a～702nが選択され、選択された署名モジュールに生成した公開鍵証明書を送信するとともに署名実行命令を送信する。

## 【0146】

公開鍵証明書と署名実行命令を受信した各署名モジュールは、それぞれのモジュールによって実行可能な署名アルゴリズム（ex. RSA, ECC）に従って署名処理を実行し、署名が実行された公開鍵証明書をCAサーバ41に返送する。CAサーバ701は、各モジュールにおいて署名のなされた公開鍵証明書を各モジュールから受信して発行要求を出した登録局（RA）751～755に送付する。

## 【0147】

署名モジュール702a～702nの各々は、署名を実行するための署名アルゴリズムに従った認証局の署名鍵を外部から入力して格納するかまたは自ら生成し、その署名鍵を用いて署名を実行する。署名モジュール702a～702nの各々は、署名実行用の専用ハードウェア（HSM: Hardware Security Module）、あるいは署名アルゴリズムを実行可能なプログラムによって署名を実行する専用プロセッサ、あるいはCPUを備えたモジュールとして構成される。署名モジュール702a～702nの各々は耐タンパー性を持ち、CAサーバ701の生成した公開鍵証明書の構成要素に基づくメッセージに対して署名鍵による署名処理を実行する。なお、以下の説明では、署名モジュールを持つ処理部をHSMとして説明するが、HSMは、署名アルゴリズムを実行可能なプログラムによって署名を実行する専用プロセッサ、あるいはCPUを備えたソフトウェアによる署名

処理を実行するモジュールによっても置き換え可能である。

#### 【0148】

署名モジュール702a～702nの各々の実行する署名処理の例として、前述したRSA (Rivest-Shamir-Adleman) 暗号方式、楕円曲線 (ECC: Elliptic Curve Cryptography) 暗号方式、デジタル署名方式 (DSS: Digital Signature Standard) などがあり、さらに、各方式において適用する鍵長により、演算速度、セキュリティ度合いなどが異なる。鍵長としては、例えば現在使われているものとして、RSA暗号: 512bit、1024bit、2048bit、また、楕円曲線暗号 (ECC) 方式では、160bit、192bit、224bitがある。また、楕円曲線暗号については、体 $F(p)$  ( $p$ は素数または2のべき乗) 上の楕円曲線 $y^2=x^3+ax+b$ において、体の標数 $p$ 、位数  $r$ 、 $a$ 、 $b$ 、曲線上のベースポイント $Gx, Gy$  によって署名付けを行うアルゴリズムが決定しセキュリティ強度も異なる。

#### 【0149】

このように複数の署名アルゴリズムを実行可能な認証局を構成することにより、複数署名付き公開鍵証明書を単一の認証局において作成可能となり、複数の認証局を経由させる必要がなくなる。

#### 【0150】

##### [デバイス (情報処理装置) 構成]

次に、認証局 (CA) の発行した公開鍵証明書の利用主体であるエンドエンティティ (EE) のデバイス (情報処理装置) において、複数の署名アルゴリズムの処理 (検証) を可能とした構成について図22、図23を用いて説明する。

#### 【0151】

図22は、エンドエンティティ (EE) のデバイス構成のブロック図である。図22に示すように、デバイスは、他のデバイス、コンテンツプロバイダ、あるいは登録局 (RA) 等との通信を実行する通信部831、デバイス全体のデータ入出力などの制御を実行する上位コントローラ832、マウス、キーボードなどから構成される入力手段833、CRT、LCDなどの表示手段834、署名検証、認証、暗号化、復号処理を実行する暗号処理部810、コンテンツの暗号化

、復号処理などに適用する鍵情報などを格納する外部メモリ 835、自デバイスの公開鍵証明書、サービスプロバイダの公開鍵証明書、暗号化コンテンツ等を格納する大容量記憶部 836 を有する。

#### 【0152】

暗号処理部 810 は、暗号処理部全体を制御する制御部 811、デバイスの識別子 (ID)、機器固有の秘密鍵などの情報を格納する記憶モジュール 812、相互認証処理プログラムを実行する相互認証モジュール 813、該部メモリ 835 のアクセス制御を実行する外部メモリ制御部 814、暗号化、復号処理を実行する暗号／復号化処理モジュール 820 を有する。

#### 【0153】

暗号／復号化処理モジュール 820 は、コンテンツ、データの復号処理を実行する復号化ユニット 821、コンテンツ、データの暗号化処理を実行する暗号化ユニット 822、暗号化処理、署名処理、鍵生成などの実行時に用いる乱数を発生する処理を実行する乱数発生ユニット 823、暗号化処理、署名処理、鍵生成などにおけるハッシュ計算を行なうハッシュ計算ユニット 824、さらに、RSA 署名生成を実行する RSA 署名生成ユニット 825、RSA 署名検証を実行する RSA 署名検証ユニット 826、ECC 署名生成を実行する ECC 署名生成ユニット 827、ECC 署名検証を実行する ECC 署名検証ユニット 828 を有する。

#### 【0154】

図 23 にデバイスの各記憶手段に格納するデータを説明する図を示す。まず、大容量記憶部 836 には、デバイスに固有の公開鍵証明書が格納されている。この公開鍵証明書は、例えば図 11～13 を用いて説明した認証局 (CA) が発行した公開鍵証明書であり認証局 (CA) の署名が 1 以上含まれる公開鍵証明書である。

#### 【0155】

さらに、デバイスがデータ通信を実行するサービスプロバイダ、コンテンツプロバイダ、あるいは他のデバイスの公開鍵証明書を格納する。これらも同様に認証局 (CA) が発行したものであり認証局 (CA) の署名が 1 以上含まれる公開

鍵証明書である。その他、大容量記憶部 8 3 6 には、暗号化コンテンツ、その他の登録情報などを格納する。

## 【 0 1 5 6 】

外部メモリ 8 3 5 には、コンテンツの暗号化鍵、復号鍵として例えばコンテンツキーなどが格納される。

## 【 0 1 5 7 】

暗号処理部内の記憶部 8 1 2 には、デバイスの識別子 ( I D ) 、デバイス固有の秘密鍵、その他の秘密鍵、例えば他デバイスとの共通鍵暗号方式などでの暗号化鍵として適用する秘密鍵等が格納され、さらに公開鍵証明書の検証に用いる認証局の公開鍵、サービスプロバイダの提供する暗号化データの復号などに用いるサービスプロバイダの公開鍵、さらに、該部メモリに格納したデータに関する検証用データとしてのチェックサムなどが格納される。

## 【 0 1 5 8 】

図 2 2 に示すように、デバイスは、 R S A 署名アルゴリズム、 E C C 署名アルゴリズムの両アルゴリズムの処理が可能な構成である。従って、他デバイス、あるいはサービスプロバイタとの相互認証処理、暗号化データ通信の際に通信相手から送付される公開鍵証明書が E C C 、 R S A いずれの署名アルゴリズムに従った署名を有する構成であっても、署名検証処理が実行可能である。

## 【 0 1 5 9 】

このように、複数の署名アルゴリズムの処理 ( 検証 ) が可能なデバイス ( 情報処理装置 ) を構成することにより、前述の複数署名付き公開鍵証明書以外の従来型の単一署名付き公開鍵証明書についても、ほとんどの場合、検証処理が可能となり、様々なデバイス、プロバイダ間との相互認証、暗号化通信が公開鍵証明書の検証のもと安全に実行可能となる。

## 【 0 1 6 0 】

以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に

記載した特許請求の範囲の欄を参酌すべきである。

【 0 1 6 1 】

【発明の効果】

上述したように、本発明の公開鍵証明書発行システム、公開鍵証明書発行方法、および情報処理装置、情報記録媒体、並びにプログラム記憶媒体によれば、RSA、ECCなど様々な署名方式に従った複数の署名を格納した公開鍵証明書を発行し、デバイス側で自デバイスにおいて処理（検証）可能な署名を選択して検証処理を実行する構成としたので、異なる署名アルゴリズムのみを検証可能なデバイス間においても相手方の公開鍵証明書の検証処理が可能となり、自デバイスと同様の特定の署名アルゴリズムの署名の付加された公開鍵証明書を持つデバイスのみに限らず、自デバイスと異なる署名アルゴリズムによる署名を持つ公開鍵証明書を有するデバイス、あるいはプロバイダとの相互認証、暗号データ通信における公開鍵証明書の検証が実行でき、通信における信頼性を高めることが可能となる。

【図面の簡単な説明】

【図 1】

一般的な公開鍵証明書の例を示す図である。

【図 2】

従来の公開鍵証明書の発行システムの概要を説明する図である。

【図 3】

従来の公開鍵証明書の発行システムにおいて発行された公開鍵証明書を使用した認証処理概要を説明する図である。

【図 4】

本発明の公開鍵証明書発行システムの概要を説明する図である。

【図 5】

公開鍵証明書のデータ構成の詳細を説明する図（その 1）である。

【図 6】

公開鍵証明書のデータ構成の詳細を説明する図（その 2）である。

【図 7】

ECC署名生成処理の手順を説明するフロー図である。

【図 8】

ECC署名検証処理の手順を説明するフロー図である。

【図 9】

RSA署名処理に必要な鍵の生成手順を説明するフロー図である。

【図 1 0】

RSA署名生成処理、検証処理の手順を説明するフロー図である。

【図 1 1】

公開鍵証明書構成および署名処理例（その 1）を示す図である。

【図 1 2】

公開鍵証明書構成および署名処理例（その 2）を示す図である。

【図 1 3】

公開鍵証明書構成および署名処理例（その 3）を示す図である。

【図 1 4】

公開鍵証明書の発行および署名処理手順を説明するフロー図（例 1）である。

【図 1 5】

公開鍵証明書の発行および署名処理手順を説明するフロー図（例 2）である。

【図 1 6】

公開鍵証明書の署名検証処理手順を説明するフロー図（例 1 - 1）である。

【図 1 7】

公開鍵証明書の署名検証処理手順を説明するフロー図（例 1 - 2）である。

【図 1 8】

公開鍵証明書の署名検証処理手順を説明するフロー図（例 2 - 1）である。

【図 1 9】

公開鍵証明書の署名検証処理手順を説明するフロー図（例 2 - 2）である。

【図 2 0】

異なる署名検証処理を実行するデバイス間での相互認証処理を説明する図である。

【図 2 1】



異なるアルゴリズムによる複数署名を生成可能な認証局構成を示すブロック図である。

【図 2 2】

異なるアルゴリズムによる署名検証可能なデバイス構成を示すブロック図である。

【図 2 3】

異なるアルゴリズムによる署名検証可能なデバイスの記憶手段の格納データを説明する図である。

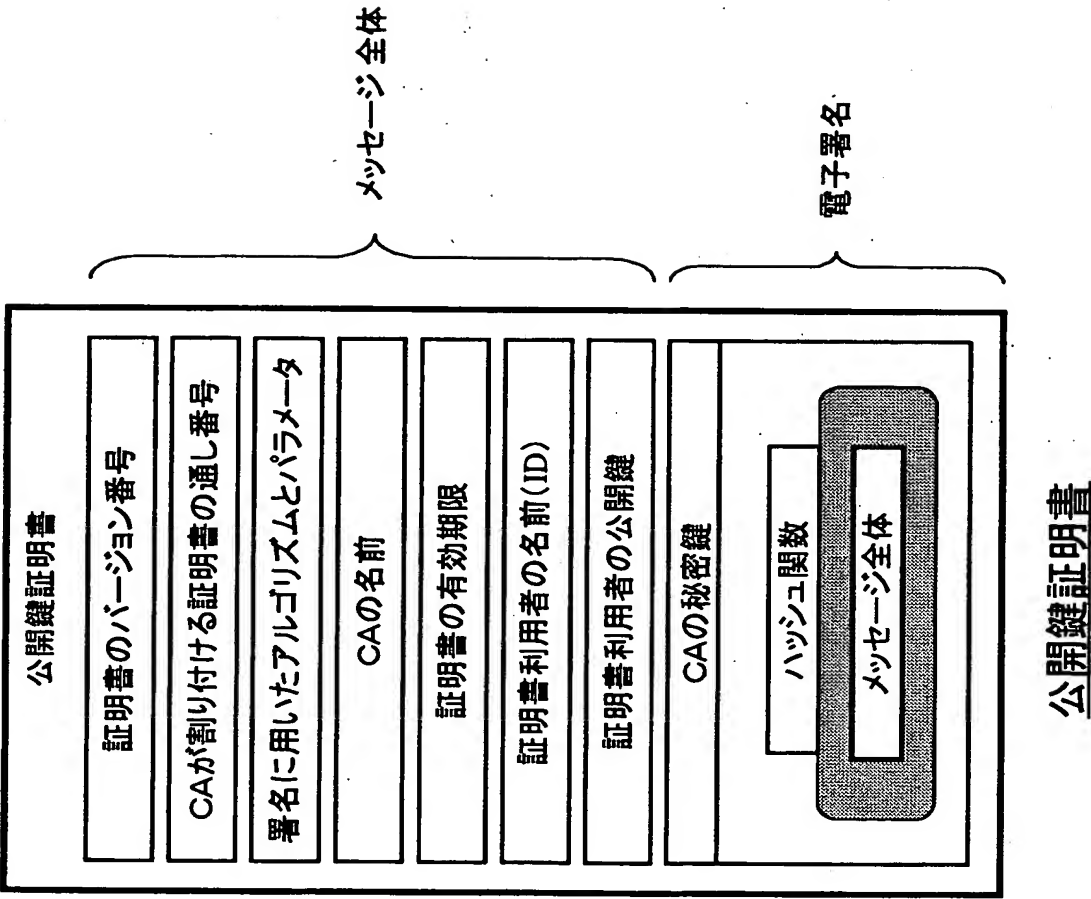
【符号の説明】

- 2 1    ECC 認証局
- 2 2    ECC 登録局
- 2 3    ECC デバイス
- 3 1    RSA 認証局
- 3 3    RSA デバイス
- 4 1    RSA デバイス
- 4 2    ECC デバイス
- 4 3    RSA 認証局
- 4 4    ECC 認証局
- 4 5    RSA & ECC 登録局
- 6 1, 7 1, 8 1    基本領域
- 6 2, 7 2, 8 2    拡張領域
- 6 3, 7 3, 8 3    署名領域
- 6 4, 7 4, 8 4    署名情報領域
- 7 5, 8 5    第 2 署名領域
- 7 6, 8 6    フラグ
- 7 7, 8 7    署名情報
- 5 0 1    RSA デバイス
- 5 0 2    ECC デバイス
- 5 0 3    RSA 認証局

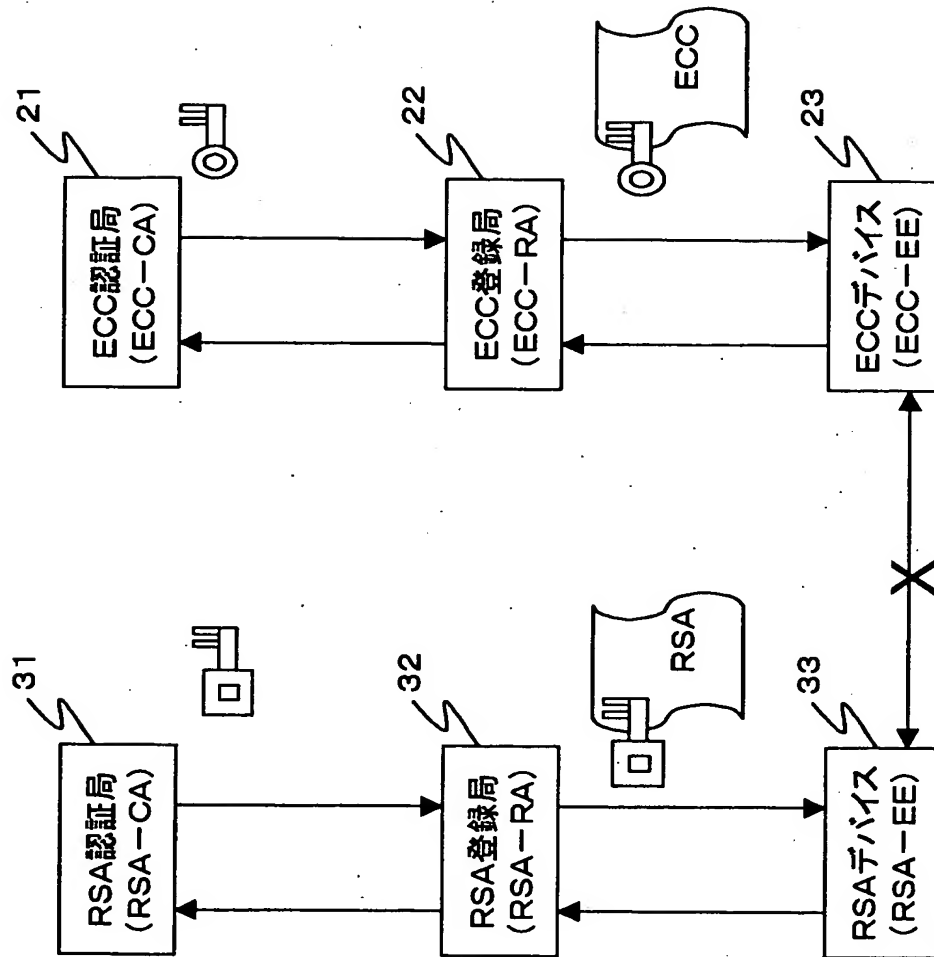
- 5 0 4    E C C 認 証 局
- 5 0 5    R S A & E C C 登 録 局
- 7 0 0    認 証 局 ( C A )
- 7 0 1    C A サ ー バ
- 7 0 2 a ~ 7 0 2 n    署 名 モ ジ ュ ー ル
- 7 5 1 ~ 7 5 5    登 録 局 ( R A )
- 8 1 0    暗 号 処 理 部
- 8 1 1    制 御 部
- 8 1 2    記 憶 モ ジ ュ ー ル
- 8 1 3    相 互 認 証 モ ジ ュ ー ル
- 8 1 4    外 部 メ モ リ 制 御 部
- 8 2 0    暗 号 / 復 号 化 処 理 モ ジ ュ ー ル
- 8 2 1    復 号 化 ユ ニ ッ ト
- 8 2 2    暗 号 化 ユ ニ ッ ト
- 8 2 3    乱 数 発 生 ユ ニ ッ ト
- 8 2 4    ハ ッ シ ュ 計 算 ユ ニ ッ ト
- 8 2 5    R S A 署 名 生 成 ユ ニ ッ ト
- 8 2 6    R S A 署 名 検 証 ユ ニ ッ ト
- 8 2 7    E C C 署 名 生 成 ユ ニ ッ ト
- 8 2 8    E C C 署 名 検 証 ユ ニ ッ ト
- 8 3 1    通 信 部
- 8 3 2    上 位 コ ン ト ロ ー ラ
- 8 3 3    入 力 手 段
- 8 3 4    表 示 手 段
- 8 3 5    外 部 メ モ リ
- 8 3 6    大 容 量 記 憶 部

【書類名】                      図面

【図 1】

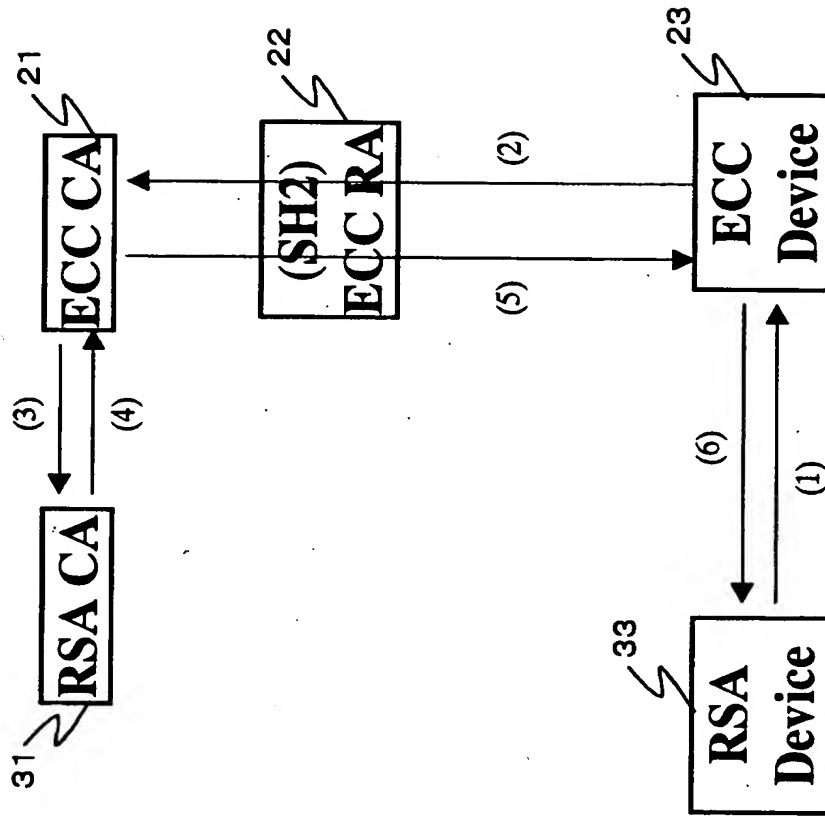


【図 2】

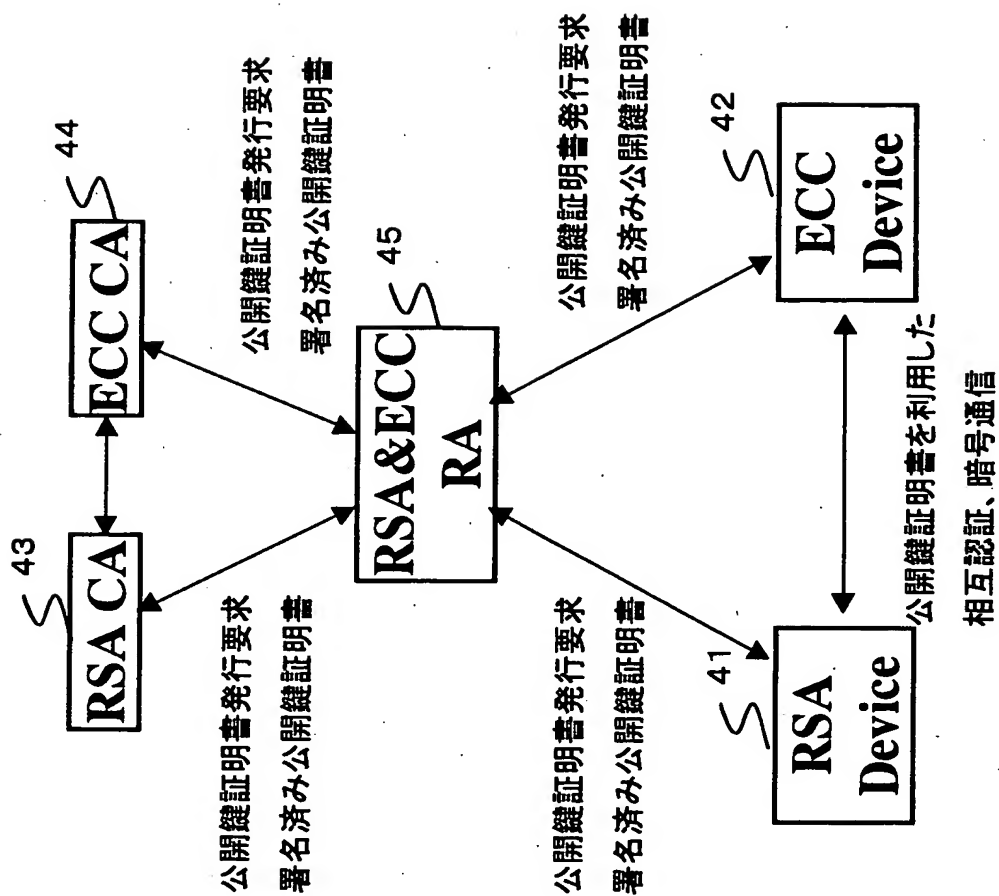


【図 3】

異なる署名方式を利用するEE同士の相互認証時



【圖 4】



【図 5】

証明書フォーマット例 (X.509 V3に準拠)

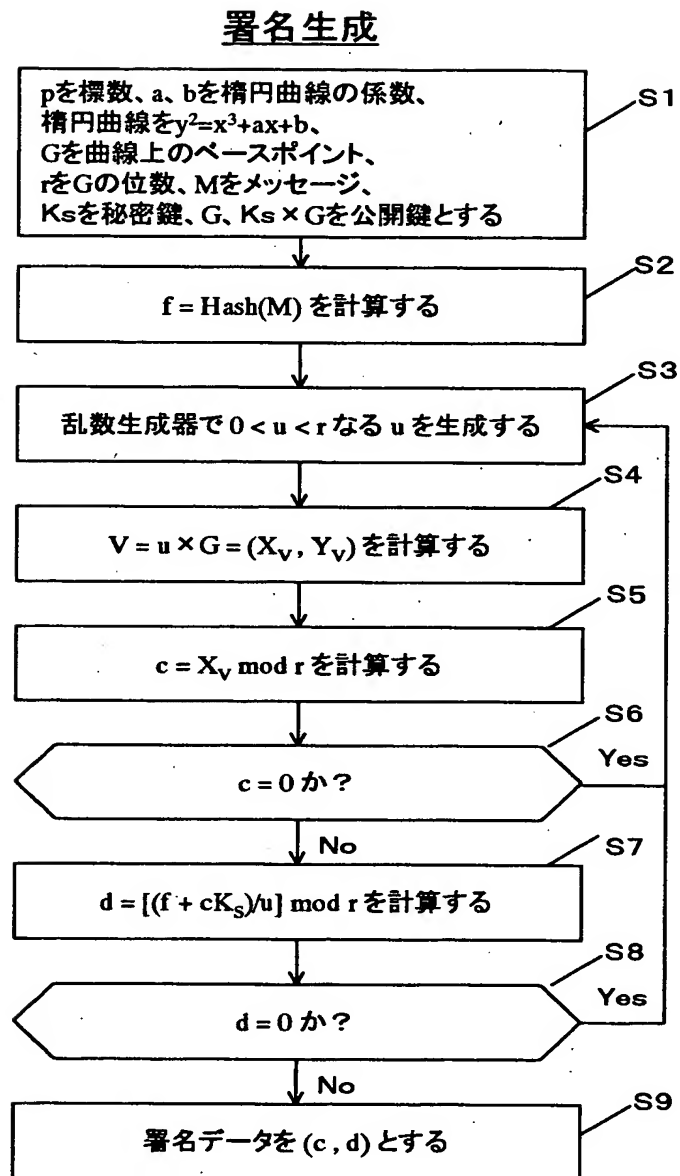
| 項目   | 説明  | 本 IA における設定                              |
|--|---|--|
| Version1   |   |  |
| version  | 証明書のフォーマットのバージョン  | V3                                       |
| serial Number  | IA によってつけられる証明書の Serial No.   | シーケンシャルなシリアルナンバー                         |
| signature.algorithm Identifier<br>algorithm parameters   | 証明書の署名アルゴリズム、及びそのパラメータ  | 楕円曲線暗号/RSA<br>楕円の場合パラメータ<br>RSAの場合鍵長     |
| issuer   | IA 名 (Distinguished Name 形式)  | 本 IA の名称                                 |
| validity<br>notBefore<br>notAfter  | 証明書の有効期限<br>開始日時<br>終了日時  |  |
| subject  | user を識別する名前  | ユーザ機器 ID またはサービス主体の ID                   |
| subject Public Key Info<br>algorithm<br>subject Public key   | user の公開鍵情報<br>鍵のアルゴリズム<br>鍵  | 楕円曲線/RSA<br>user の公開鍵                    |
| Version3   |   |  |
| authority Key Identifier<br>key Identifier<br>authority Cert Issuer<br>authority Cert Serial Number  | IA の署名確認用の鍵識別<br>鍵識別番号 (8 進数)<br>IA 名 (General Name 形式)<br>認証番号   |  |
| subject key Identifier   | 複数の鍵の証明をする場合  | 利用しない                                    |
| key usage<br>(0)digital Signature<br>(1)non Repudiation<br>(2)key Encipherment<br>(3)data Encipherment<br>(4)key Agreement<br>(5)key CertSign<br>(6)cRL Sign | 鍵の使用目的を指定<br>(0)デジタル署名用<br>(1)否認防止用<br>(2)鍵の暗号化用<br>(3)メッセージの暗号化用<br>(4)共通鍵配送用<br>(5)認証の署名確認用<br>(6)失効リストの署名確認用 | 0,1,4,6 を利用                              |
| private Key Usage Period<br>notBefore<br>notAfter  | user に格納されている秘密鍵の有効期限。  | 証明書の有効期限 = 公開鍵の有効期限 = 秘密鍵の有効期限 (default) |

【図 6】

|   |  |                                 |
|---|--|---------------------------------|
| policy Mappings<br>issuer Domain Policy<br>subject Domain Policy                                | CA を認証する場合にのみ必要。発行認証局のポリシーと被認証ポリシーのマッピングを規定                                      | default = なし                    |
| supported Algorithms<br>algorithm Identifier<br>intended Usage<br>intended Certificate Policies | ディレクトリ (X.500) のアトリビュートを定義。コミュニケーションの相手がディレクトリ情報を利用する場合に、事前にそのアトリビュートを知らせるのに用いる。 | default = なし                    |
| subject Alt Name  | user の別名 (GN 形式)。  | 利用しない                           |
| issuer Alt Name   | 項目は入れておく (default = なし)  | default = なし                    |
| subject Directory Attributes  | user の任意の属性。   | 利用しない                           |
| basic Constraints<br>cA<br>path Len Constraint  | 証明対象の公開鍵が認証局の署名用か、user のものかを区別   | default = user 用                |
| name Constraints<br>permitted Subtrees<br>base<br>minimum<br>maximum<br>excluded Subtrees       | 被認証者が認証局である場合 (CA 認証) にのみ使用。   | default = なし                    |
| policy Constraints<br>requireExplicitPolicy<br>inhibitPolicyMapping                             | 認証パスの残りに対する明確な認証ポリシー ID、禁止ポリシーマップを要求する制限を記述                                      |                                 |
| CRL Distribution Points   | user が証明書を利用する際に、証明書が失効していないかどうかを確認するための失効リストの参照ポイントを記述。                         | 証明書を登録したところへのポイント。失効リストは、発行元で管理 |
| 署名  | 発行者の署名   |                                 |

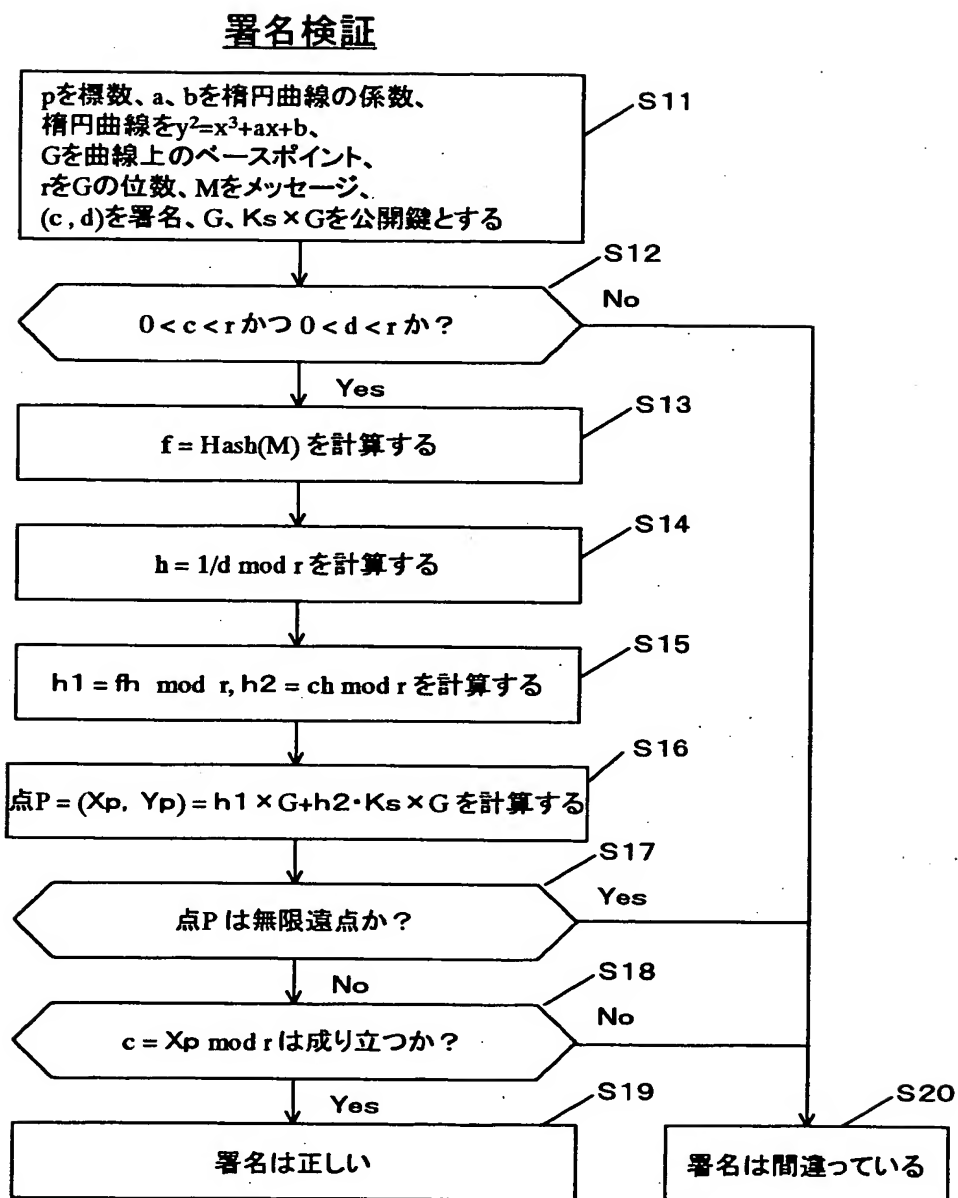


【図 7】

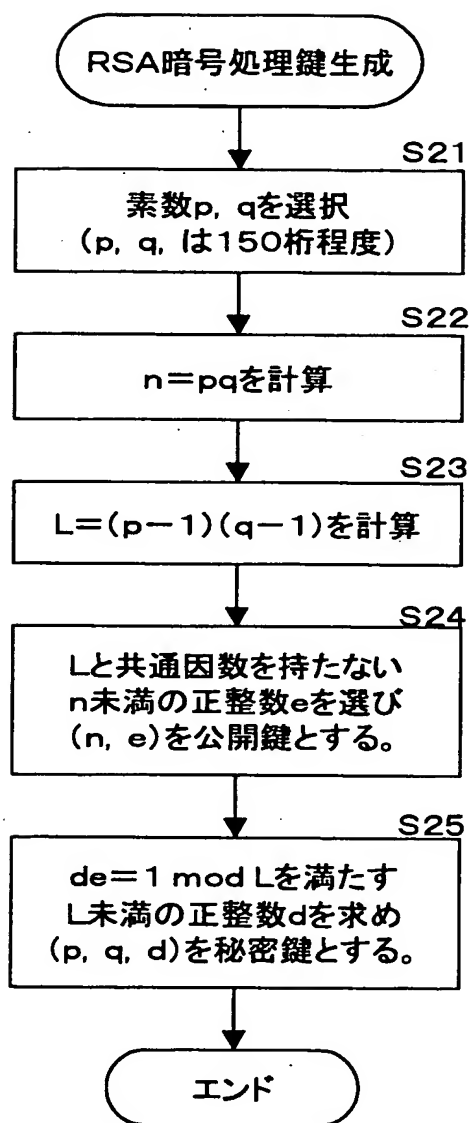


署名生成(IEEE P1363/D3)

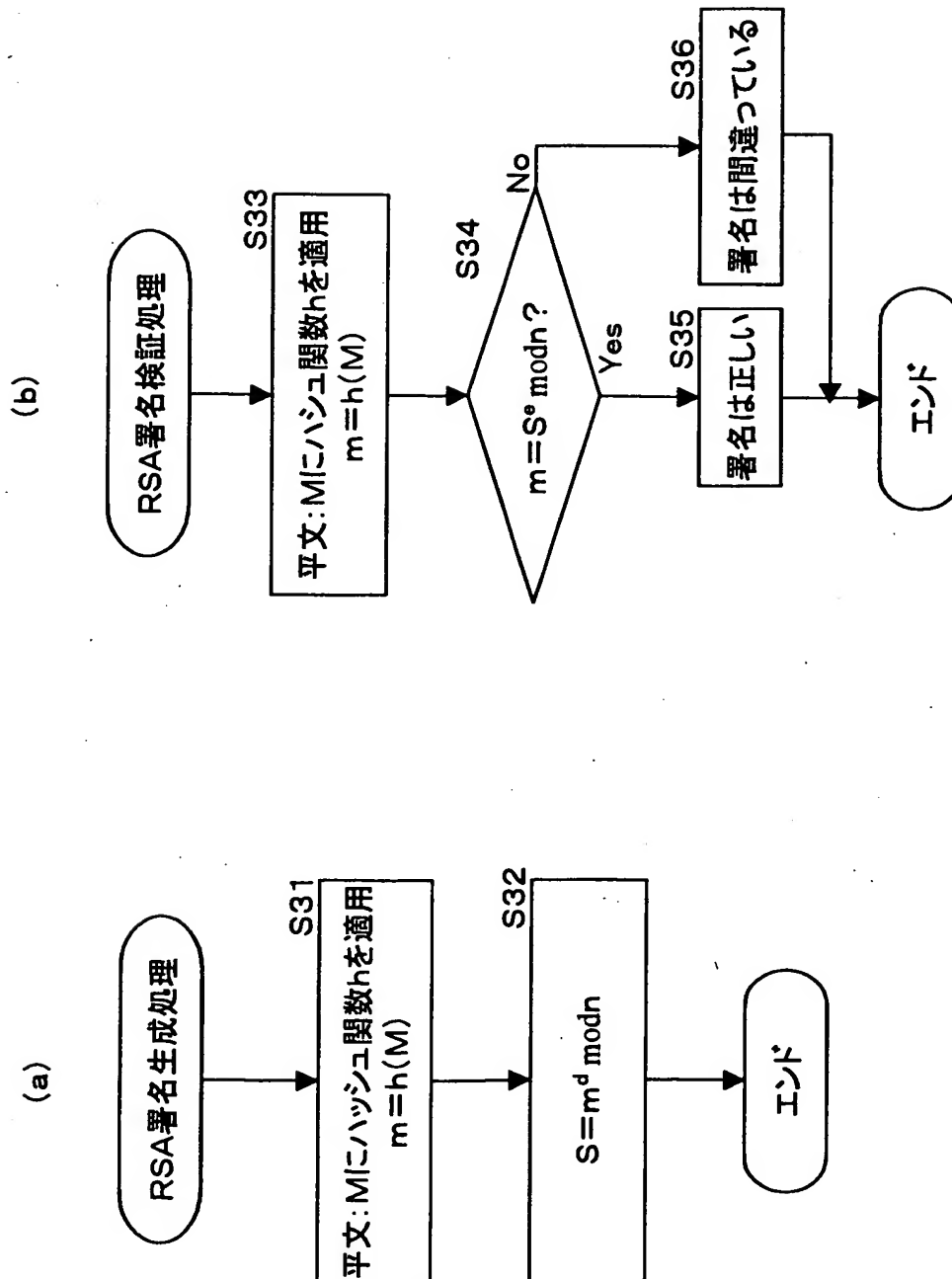
【図 8】

署名検証(IEEE P1363/D3)

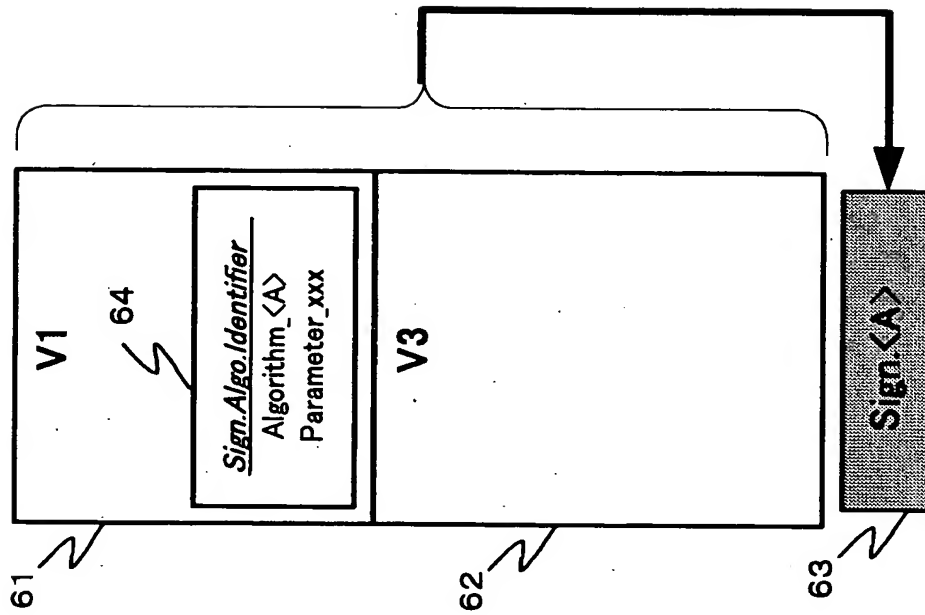
【図 9】



【図10】

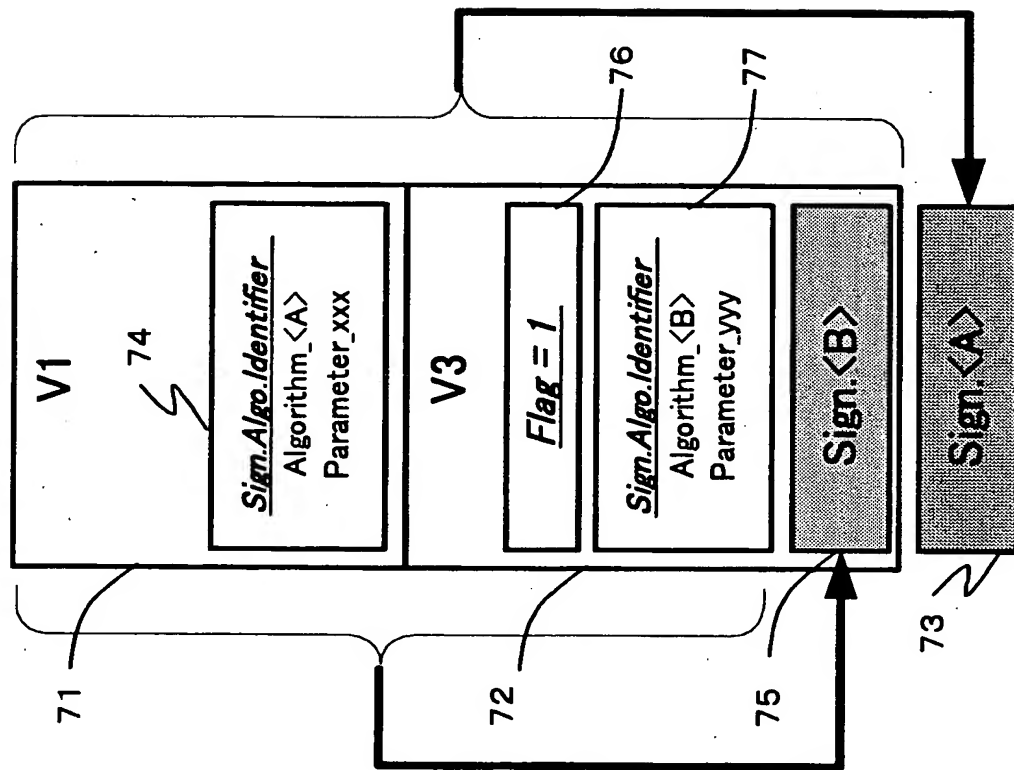


【図 11】



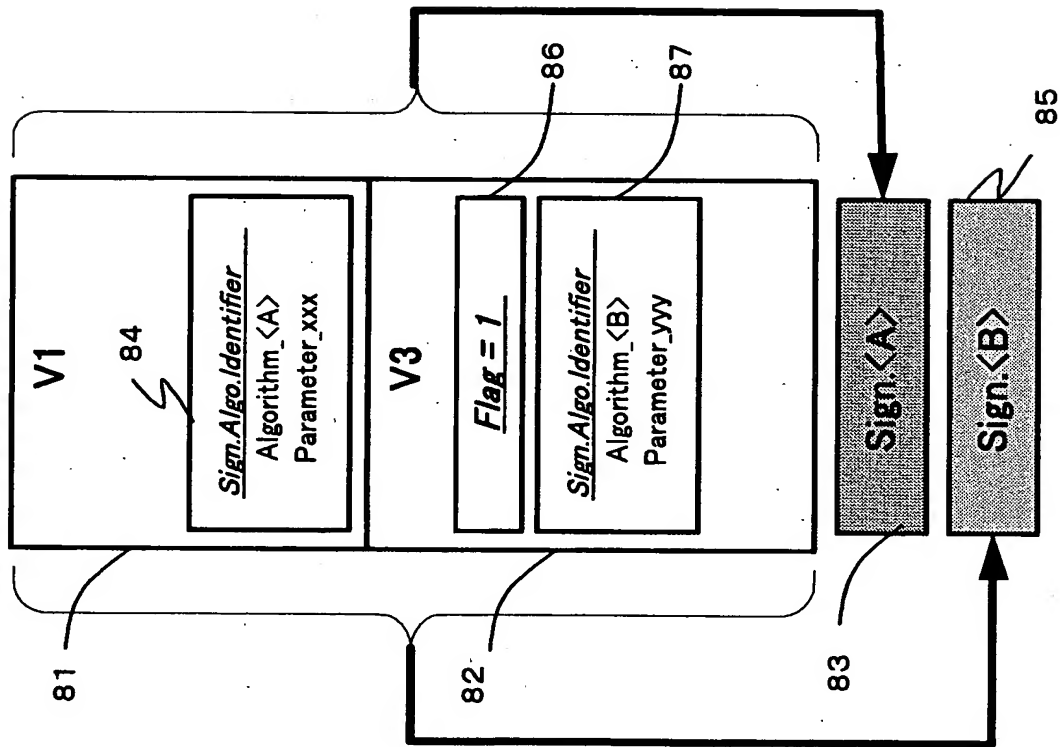
【図 12】

複数署名付証明書のフォーマット(1)



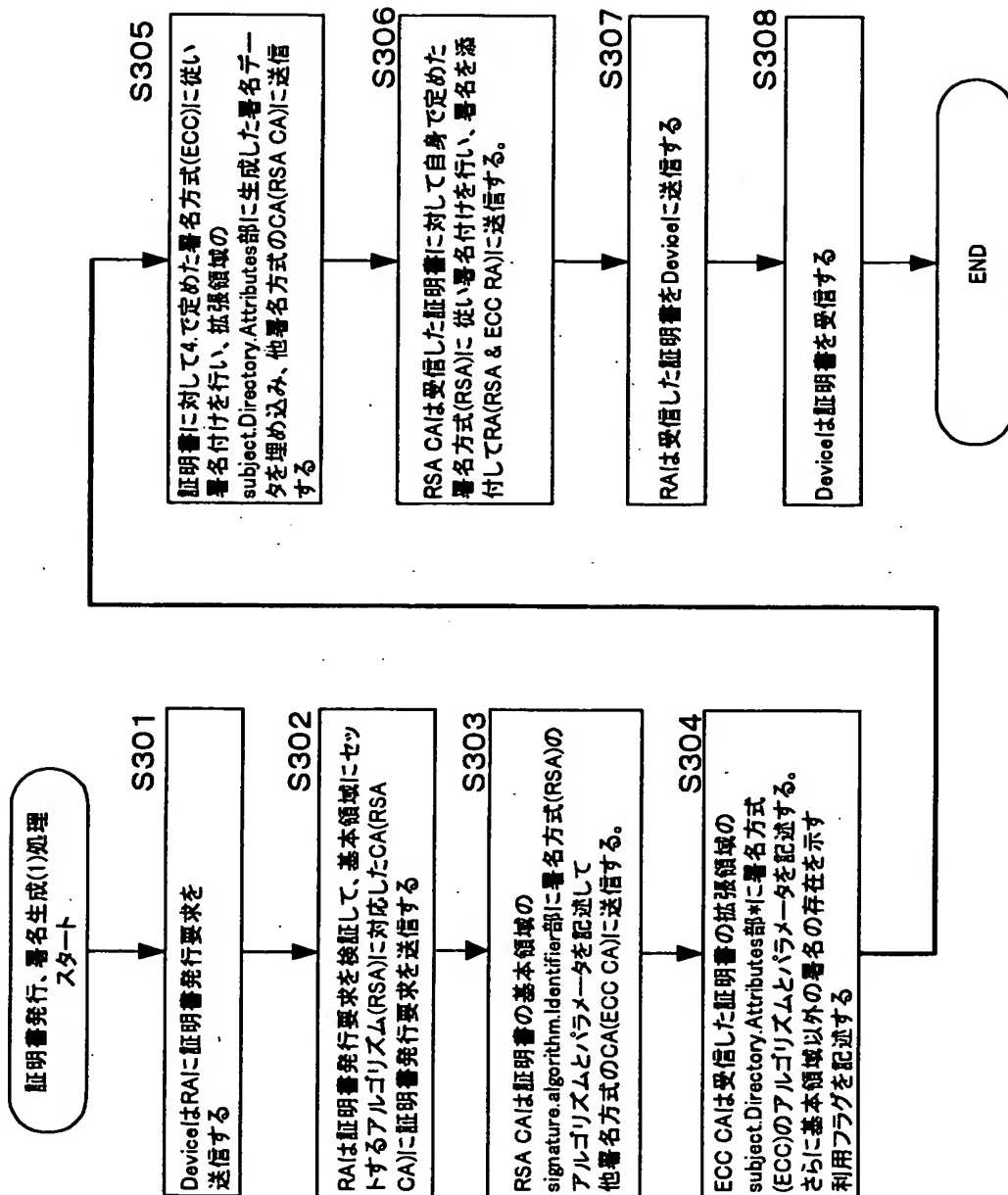
【図 13】

複数署名付証明書のフォーマット(2)



【図 1 4】

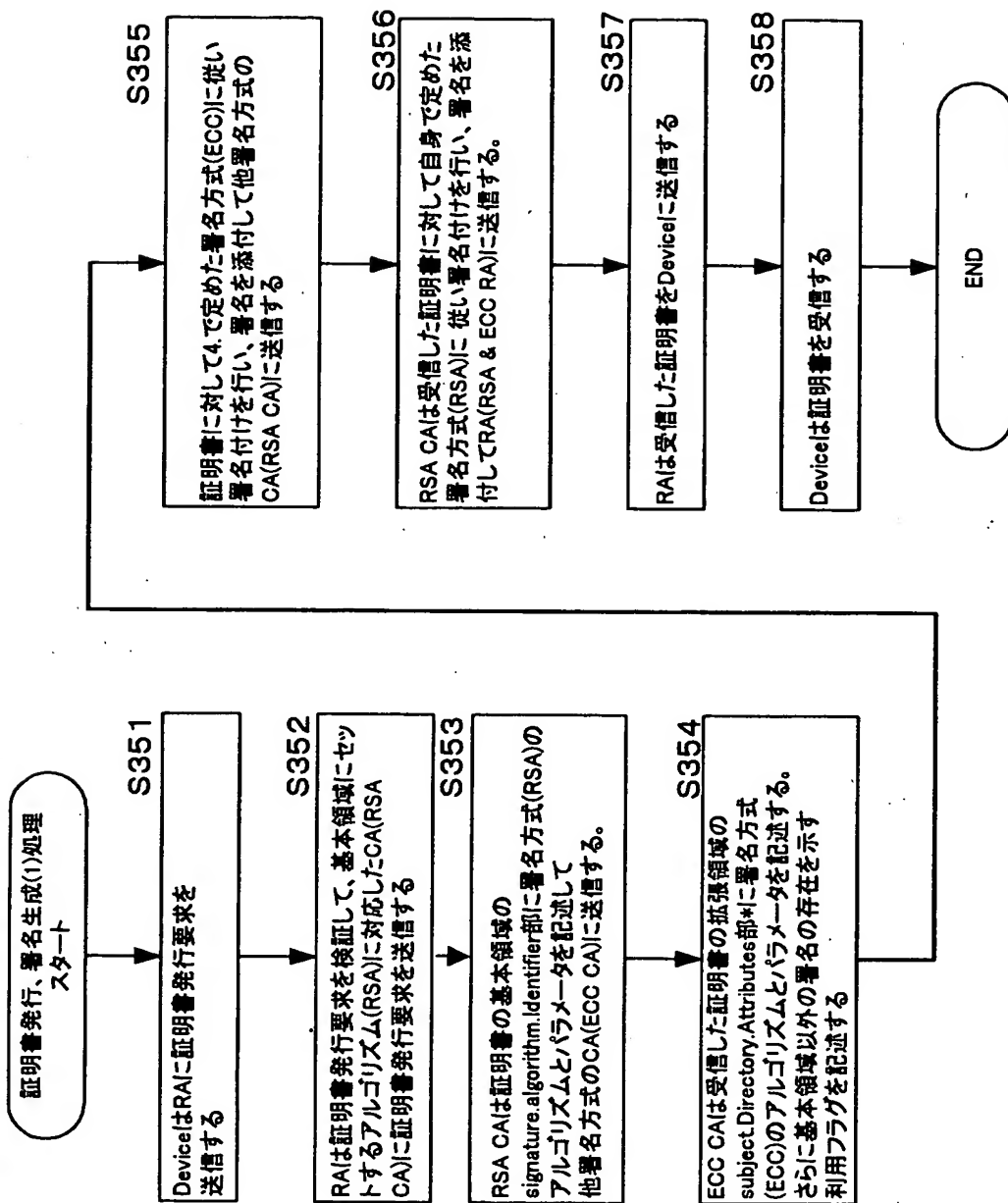
証明書発行、署名生成(1)



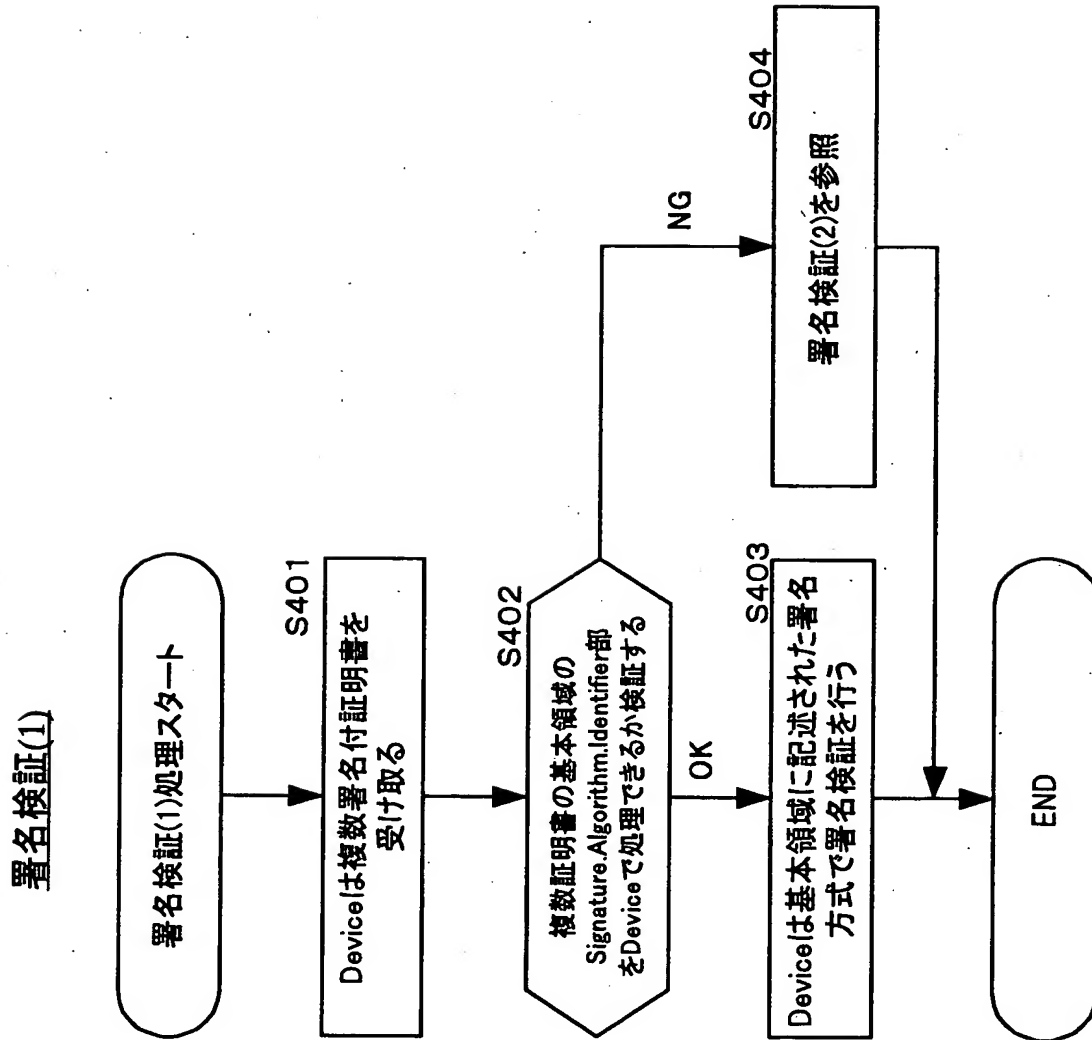


【図 1 5】

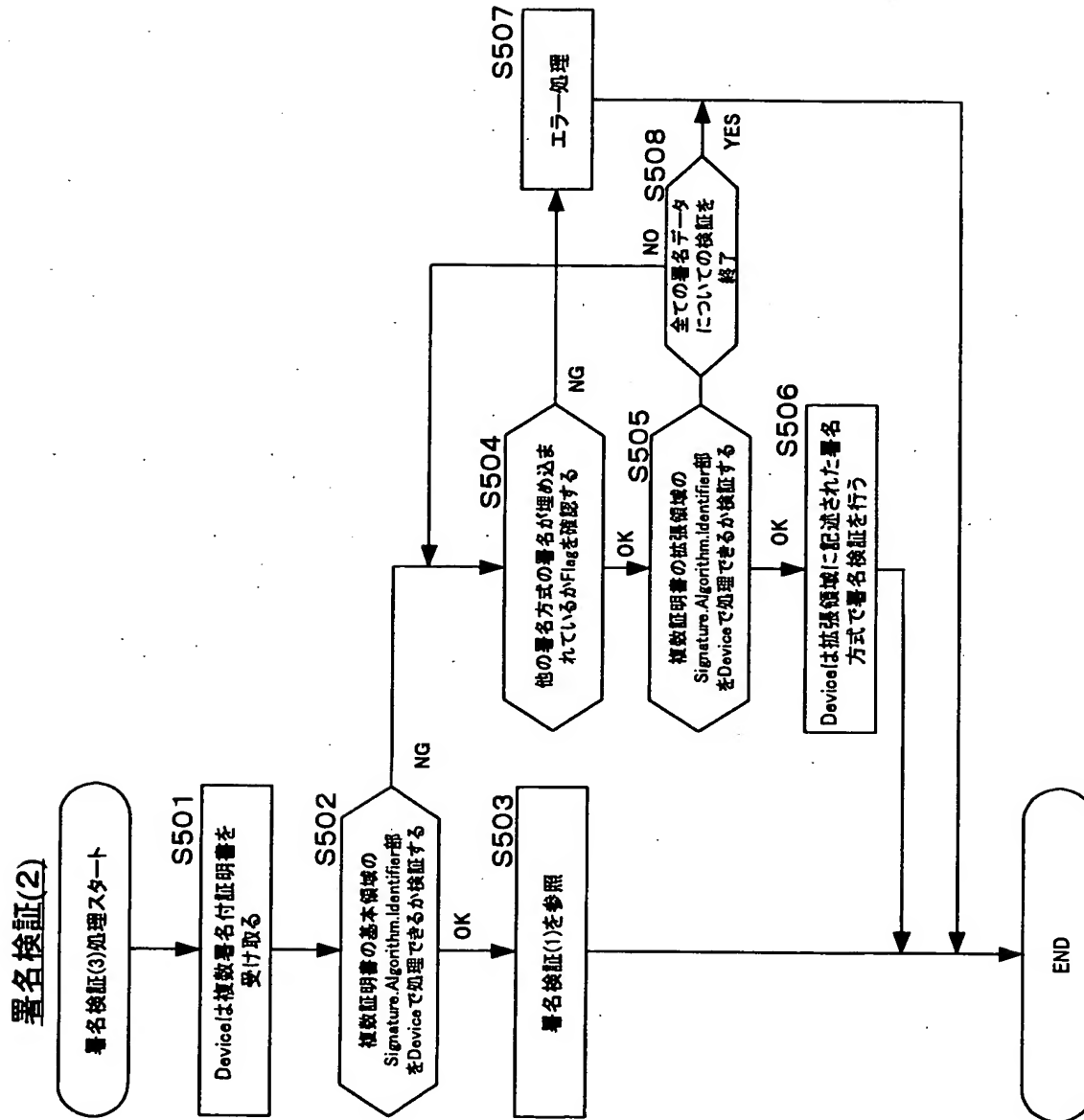
## 証明書発行、署名生成(2)



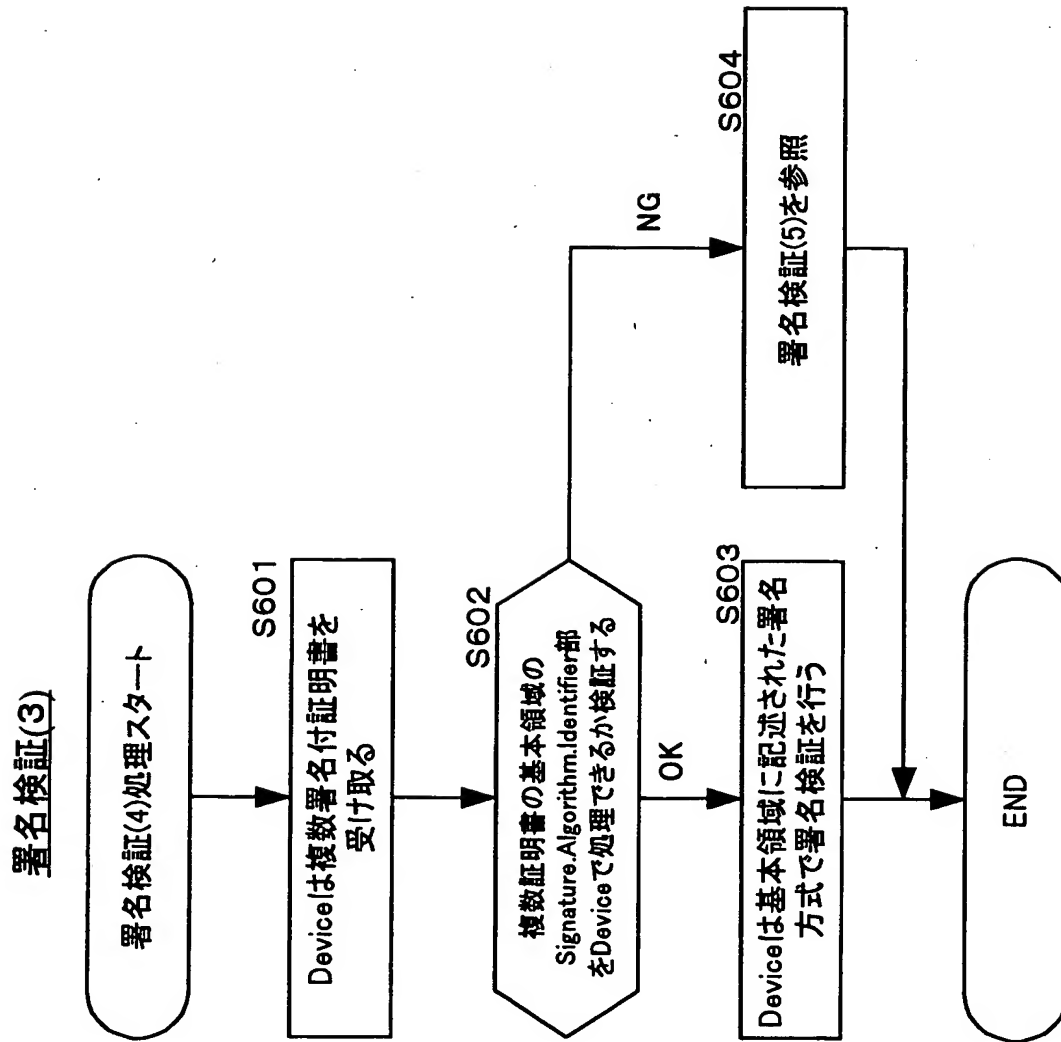
【図16】



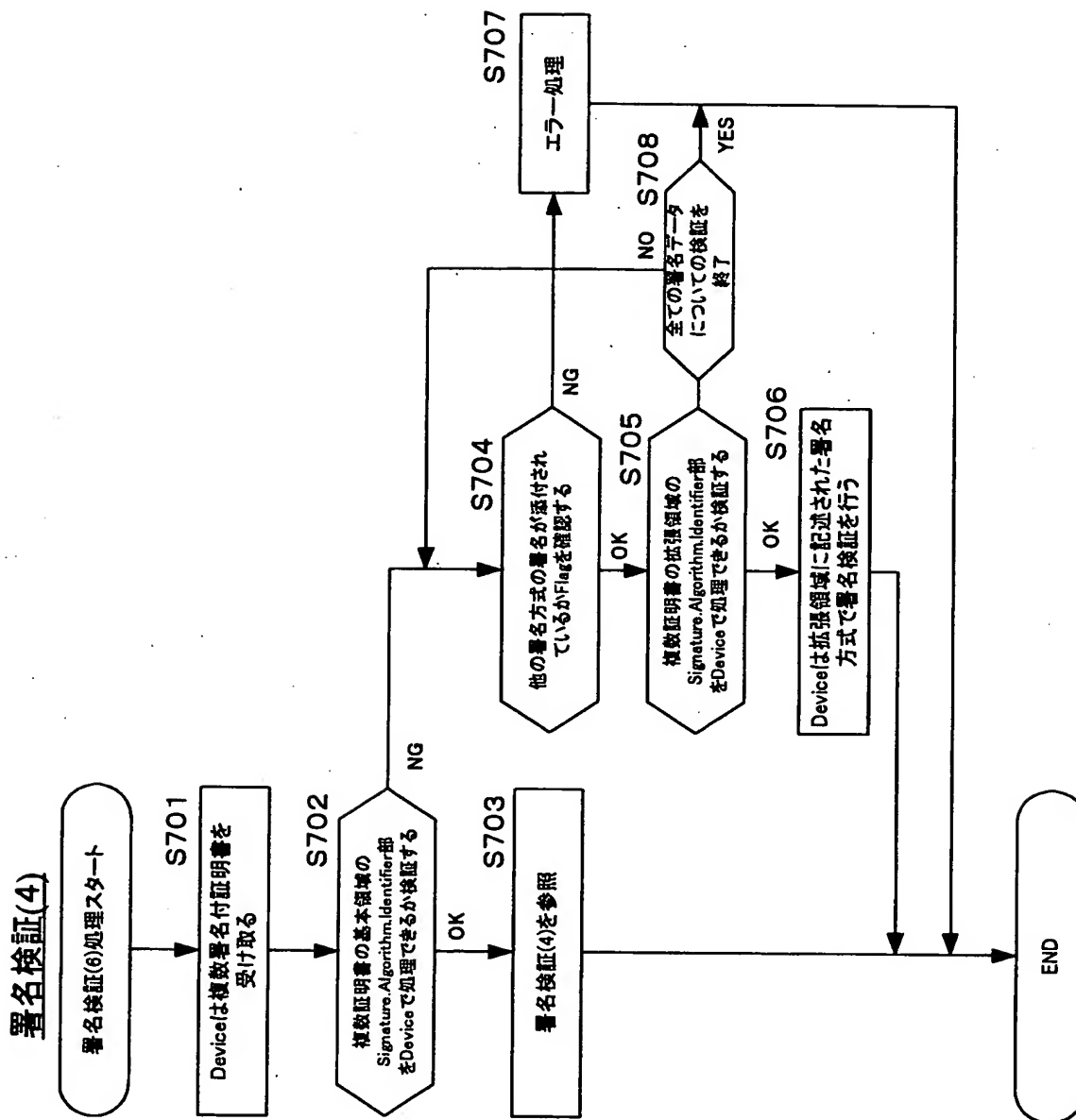
【図 17】



【図 1 8】

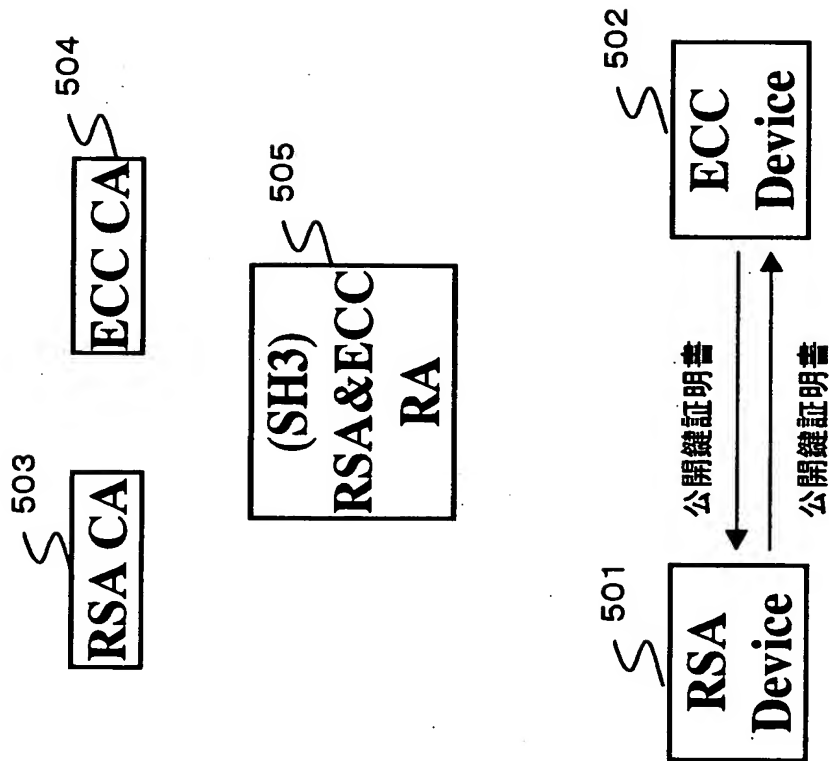


【図 19】

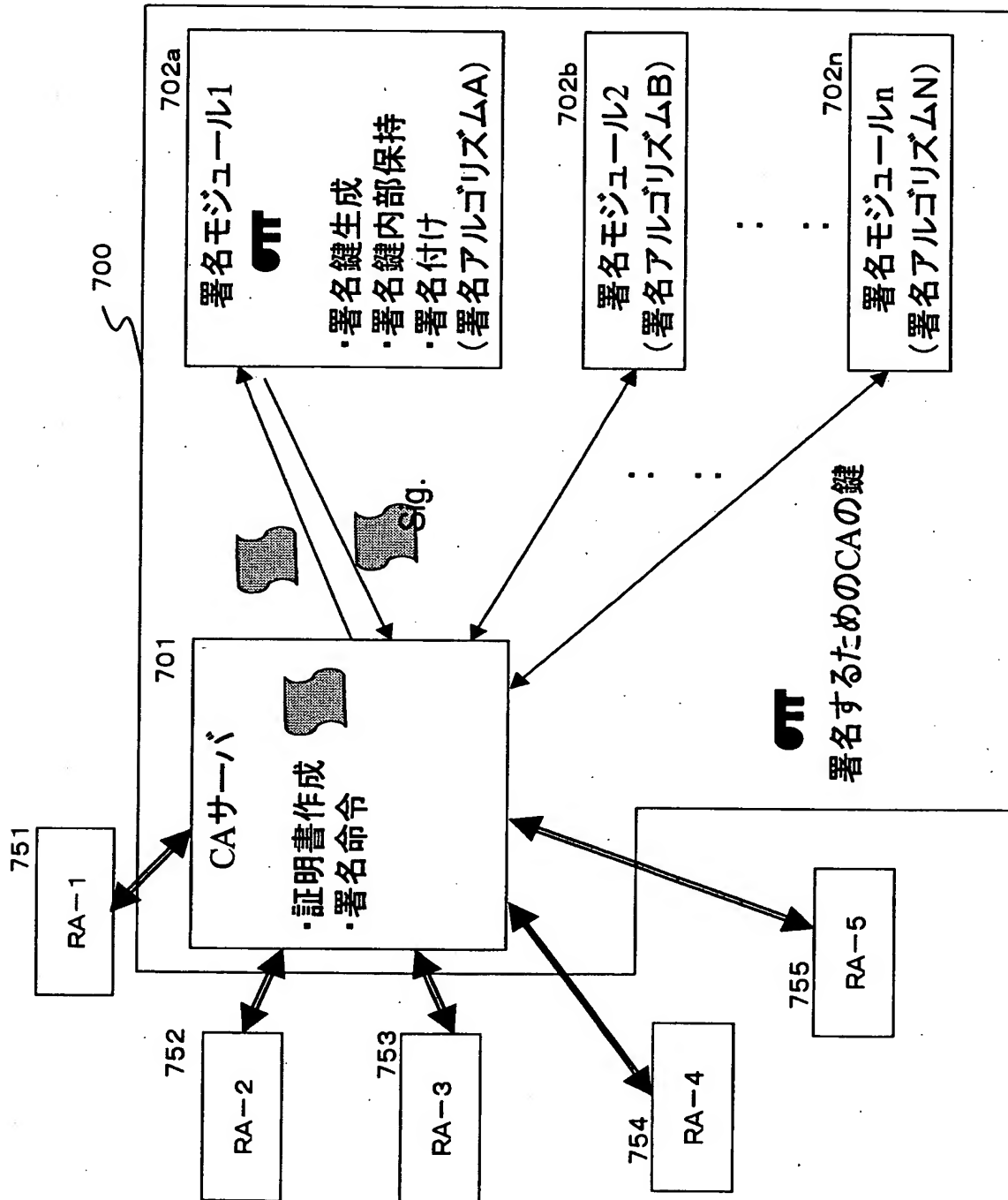


【図 2 0】

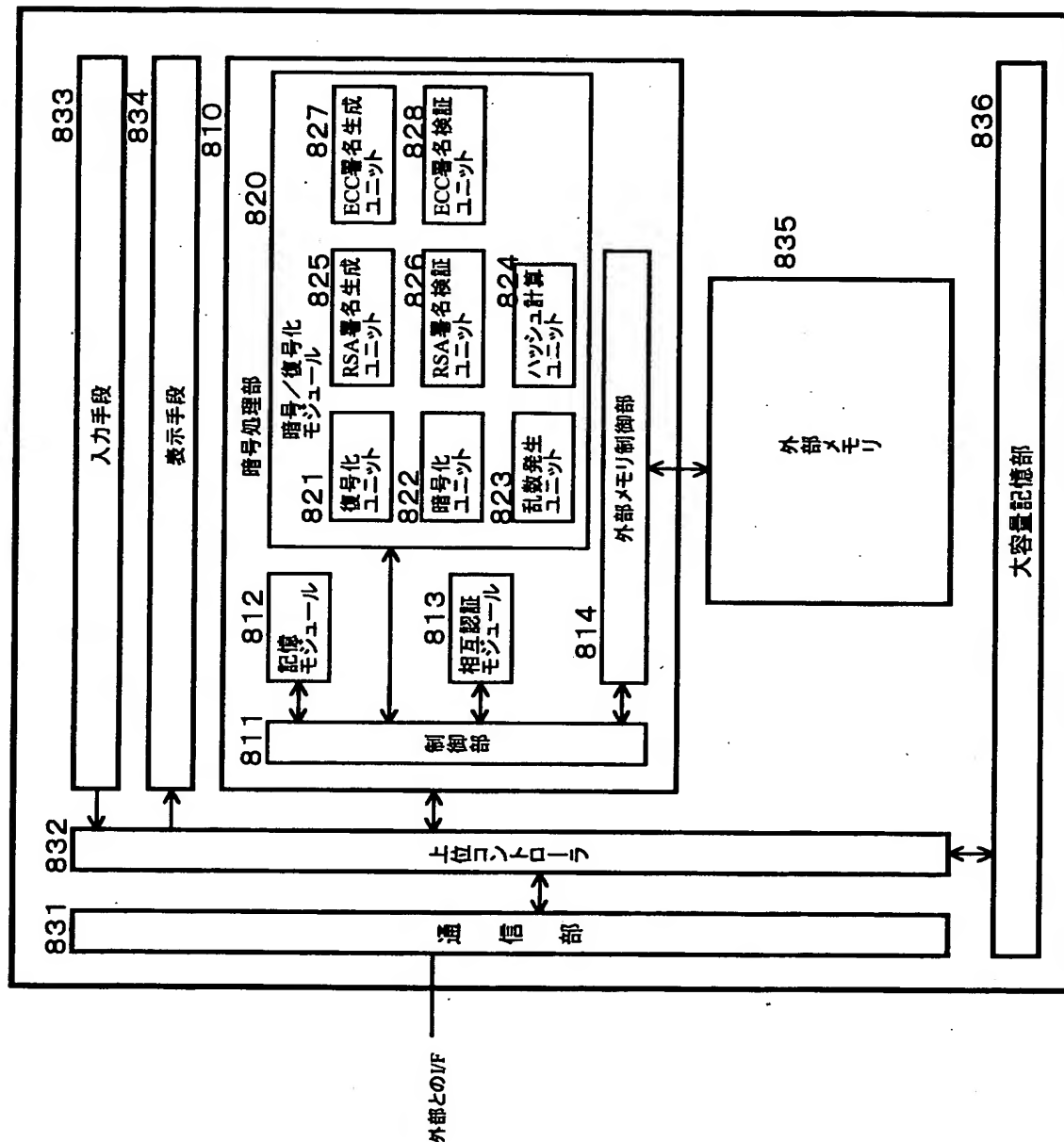
異なる署名方式を利用するEE同士の相互認証時



【図 21】



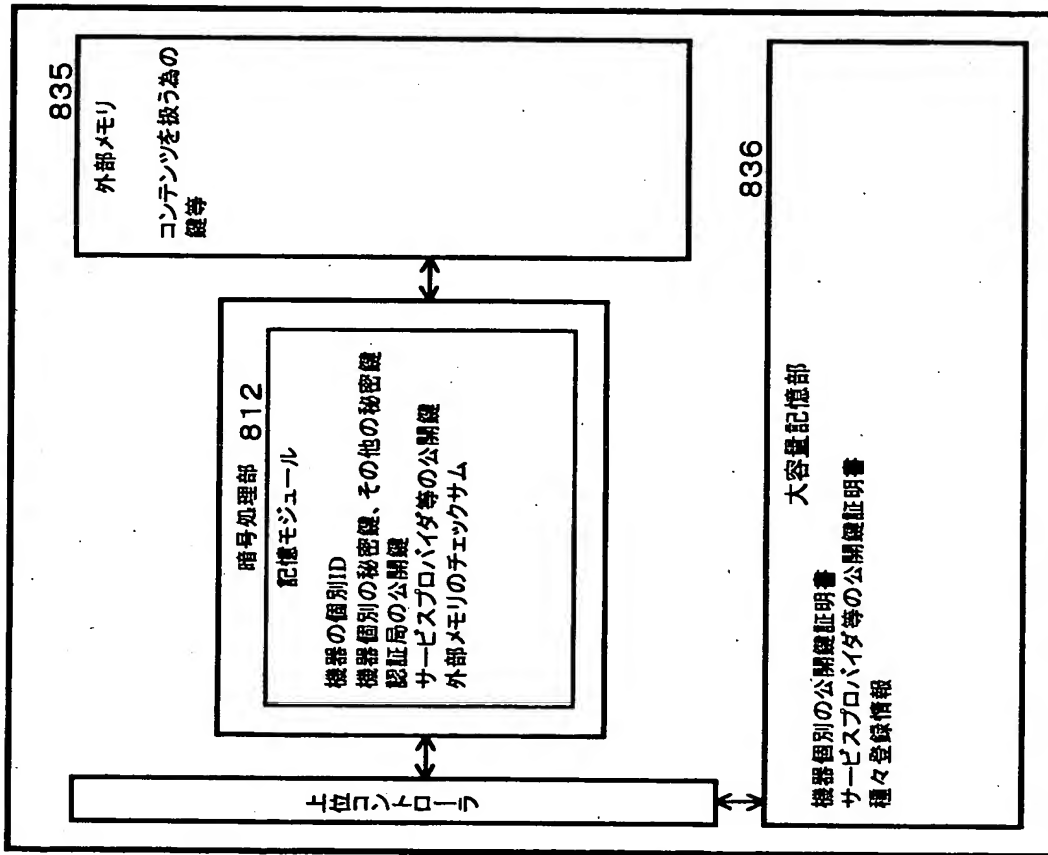
【图 2 2】





【図 23】

Deviceの保持するデータ



【書類名】 要約書

【要約】

【課題】 異なる署名アルゴリズム処理が可能なデバイス間での公開鍵証明書の検証処理を相互に可能とした構成を提供する。

【解決手段】 R S A、E C Cなど様々な署名方式に従った複数の署名を格納した公開鍵証明書を発行し、デバイス側で自デバイスにおいて処理（検証）可能な署名を選択して検証処理を実行する構成とした。従って、異なる署名アルゴリズムのみを検証可能なデバイス間においても相手方の公開鍵証明書の検証処理が可能となり、自デバイスと同様の特定の署名アルゴリズムの署名の付加された公開鍵証明書を持つデバイスのみに限らず、自デバイスと異なる署名アルゴリズムによる署名を持つ公開鍵証明書を有するデバイス、あるいはプロバイダとの相互認証、暗号データ通信における公開鍵証明書の検証が実行でき、通信における信頼性を高めることが可能となる。

【選択図】 図 1 2

出 願 人 履 歴 情 報

識別番号 [000002185]

1. 変更年月日 1990年 8月30日

[変更理由] 新規登録

住 所 東京都品川区北品川6丁目7番35号  
氏 名 ソニー株式会社